

**Analysis of microarray and next generation sequencing data for
classification and biomarker discovery in relation to complex
diseases**

A thesis submitted by

Vahid Elyasigomari

in partial fulfilment of
the requirements of the degree of

Doctor of Philosophy

in the
School of Engineering and Material Science
Queen Mary, University of London



2017

PhD Thesis Declaration

I, Vahid Elyasigomari, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

The copyright of this thesis rests with the author and no quotation from it or information derived from it may be published without the prior written consent of the author.

Signature:

Date:

Details of collaboration and publications:

- Elyasigomari, V., Mirjafari, M. J., Screen, H R C., and Shaheed, M H. (2015), Cancer classification using a novel gene selection approach by means of shuffling based on data clustering with optimization, *Journal of Applied Soft Computing*. 35(1), pp 43-51.
- Elyasigomari, V., Lee, D., Shaheed, M H. (2017), Development of a two-stage gene selection method that incorporates a novel hybrid approach using the cuckoo optimization algorithm and harmony search for cancer classification. *Journal of Biomedical Informatics*. 67(1), pp 11-20.

Abstract

This thesis presents an investigation into gene expression profiling, using microarray and next generation sequencing (NGS) datasets, in relation to multi-category diseases such as cancer. It has been established that if the sequence of a gene is mutated, it can result in the unscheduled production of protein, leading to cancer. However, identifying the molecular signature of different cancers amongst thousands of genes is complex. This thesis investigates tools that can aid the study of gene expression to infer useful information towards personalised medicine.

For microarray data analysis, this study proposes two new techniques to increase the accuracy of cancer classification. In the first method, a novel optimisation algorithm, COA-GA, was developed by synchronising the Cuckoo Optimisation Algorithm and the Genetic Algorithm for data clustering in a shuffle setup, to choose the most informative genes for classification purposes. Support Vector Machine (SVM) and Multilayer Perceptron (MLP) artificial neural networks are utilised for the classification step. Results suggest this method can significantly increase classification accuracy compared to other methods.

An additional method involving a two-stage gene selection process was developed. In this method, a subset of the most informative genes are first selected by the Minimum Redundancy Maximum Relevance (MRMR) method. In the second stage, optimisation algorithms are used in a wrapper setup with SVM to minimise the selected genes whilst maximising the accuracy of classification. A comparative performance assessment suggests that the proposed algorithm significantly outperforms other methods at selecting fewer genes that are highly relevant to the cancer type, while maintaining a high classification accuracy.

In the case of NGS, a state-of-the-art pipeline for the analysis of RNA-Seq data is investigated to discover differentially expressed genes and differential exon usages between normal and AIP positive *Drosophila* datasets, which are produced in house at Queen Mary, University of London. Functional genomic of differentially expressed genes were examined and found to be relevant to the case study under investigation. Finally, after normalising the RNA-Seq data, machine learning approaches similar to those in microarray was successfully implemented for these datasets.

Table of Contents

Abstract.....	3
Table of Contents	4
List of Figures	8
List of Tables	11
Acknowledgments	13
Chapter 1: Introduction	14
1.1: Background	14
1.2: Motivations	16
1.3: Aim and Objectives	16
1.4: Contributions.....	17
1.5: Outline of the Thesis	18
1.6: Publications	19
1.7: Utilised microarray datasets throughout this thesis	20
1.7.1: Leukaemia dataset	20
1.7.2: Lymphoma dataset.....	20
1.7.3: Prostate dataset	21
Chapter 2: Background and Recent Developments.....	22
2.1: Gene Expression	24
2.2: DNA Microarray.....	25
2.2.1: cDNA Microarrays	25
2.2.2: Oligonucleotide Microarray.....	27
2.3: Next Generation Sequencing Technology (NGS)	29
2.3.1: Roche/454 FLX Pyrosequencer	30
2.3.2: Illumina Genome Analyser	31
2.3.3: Applied Biosystems SOLiD Sequencer	34
2.3.4: NGS Raw Data File Formats and Quality Scores for Detected Nucleotides	36
2.4: Gene Expression Profiling Using NGS Technology (RNA-Seq)	38
2.5: Analytical challenges for microarray and NGS data in respect to profiling and understanding diseases	39
2.6: Summary	40

Chapter 3: Overview of Machine Learning Approaches for Microarray Data Analysis.....	41
3.1: Introduction	41
3.2: Design.....	41
3.3: Pre-Processing.....	42
3.4: Unsupervised Classification.....	44
3.4.1: <i>K-means</i>	45
3.4.2: <i>Fuzzy C-means</i>	46
3.4.3: <i>Hierarchical Clustering</i>	47
3.4.4: <i>Self-Organising Map</i>	48
3.4.5: <i>Binarisation of Consensus Partition Matrices (Bi-CoPaM)</i>	50
3.4.6: <i>Unification of clustering results from multiple datasets using external specifications (UNCLES)</i>	54
3.5: Supervised Classification	55
3.5.1: <i>Support Vector Machine</i>	55
3.5.2: <i>Multilayer Perceptron (MLP) Artificial Neural Network</i>	59
3.6: Feature Selection	64
3.7: Overfitting	66
3.8: Summary	68
Chapter 4: Effects of Data Clustering Prior to Gene Selection on Cancer Classification	69
4.1: Introduction	69
4.2: Optimisation Based Clustering Techniques.....	70
4.2.1: <i>Proposed Cost Function</i>	70
4.2.2: <i>Genetic Algorithm (GA)</i>	71
4.2.3: <i>Particle Swarm Optimisation (PSO)</i>	72
4.2.4: <i>Cuckoo Optimisation Algorithm (COA)</i>	74
4.2.5: <i>Proposed COA-GA Algorithm for Clustering</i>	76
4.3: Gene Ranking and Selection.....	77
4.4: Classification and Performance Evaluation	79
4.4.1: <i>Classification Methods</i>	79
4.4.2: <i>Performance Evaluation</i>	80
4.5: Investigating the Effects of Conventional Clustering Methods on Classification Performance	80
4.5.1: <i>Methods</i>	80
4.5.2: <i>Results</i>	82
4.6: Proposed Gene Selection Based on Shuffle Technique.....	86

4.6.1: Methods	86
4.6.2: Results	88
4.7: Summary	94
Chapter 5: Two Stage Gene Selection for Cancer Classification Using Microarray Data	96
5.1: Introduction	96
5.2: Proposed Method	98
5.3: Discretisation of Data	99
5.4: First Stage Gene Selection Using Minimum Redundancy Maximum Relevance (MRMR).....	102
5.5: Second Stage Selection Using Evolutionary Algorithms	104
5.5.1: Harmony Search Algorithm (HS).....	105
5.5.2: Proposed Algorithm COA-HS	107
5.6: Results	109
5.7: Summary	112
Chapter 6: Gene Expression Analysis using RNA-Seq Data.....	114
6.1: Overview of RNA-Seq Data Analysis.....	114
6.1.1: RNA-Seq Experimental Considerations	114
6.1.2: Pre-Processing of RNA-Seq Data	115
6.1.3: RNA-Seq Alignment	116
6.1.4: Creating a Count Table.....	118
6.1.5: Normalisation.....	118
6.1.6: Modelling Raw Counts, Dispersion and Differential Gene Expression.....	120
6.1.7: Alternative Splicing Analysis.....	124
6.2: Pipeline for Analysis of RNA-Seq: a Case Study.....	126
6.2.1: Utilised RNA-Seq Data	127
6.2.2: Pre-processing	128
6.2.3: Alignment of the Reads to a Reference Genome.....	131
6.2.4: Differential Gene Expression	133
6.2.5: Differential Exon Usage Analysis.....	143
6.2.6: Gene Annotation and Biological Relevance of Selected Genes.....	151
6.2.7: Classification	154
6.2.8: Summary	156
Chapter 7: Conclusions and Future Research	158
7.1: Analysis of Microarray Data	158

7.2: Analysis of RNA-Seq Data	160
7.3: Suggestions for Future Work.....	161
Appendix 1: R-code for differential gene expression analysis.....	162
Appendix 2: R-code for differential exon usage.....	166
Appendix 3: Differentially expressed genes	169
References	178

List of Figures

Figure 2.1: Cell structure [32].	22
Figure 2.2: Chromatin structure adapted from [33].	23
Figure 2.3: Process of transcription and translation adapted from [37].	24
Figure 2.4: Microarray glass [45].	25
Figure 2.5: DNA microarrays technology modified from [47].	26
Figure 2.6: Affymetrix GenChip lithography [50].	27
Figure 2.7: Affymetrix expression array design adapted from [51].	28
Figure 2.8: Affymetrix GenChip microarray modified from [53].	28
Figure 2.9: Sample preparation in Roche/454 [61].	30
Figure 2.10: Amplification step in Roche/454 [61].	30
Figure 2.11: Sequencing by synthesis step in Roche/454 [61].	31
Figure 2.12: Sequencing by synthesis step in Roche/454 [62].	31
Figure 2.13: Sample preparation for Illumina sequencer [64].	32
Figure 2.14: Bridge amplification for Illumina sequencer [64].	32
Figure 2.15: Sequencing by synthesis step in Illumina sequencer [64].	33
Figure 2.16: Pseudo colour enhanced image [65].	33
Figure 2.17: Outline of SOLiD sequencing technology adapted from [66].	35
Figure 2.18: FASTQ format	37
Figure 2.19: FASTA format.	37
Figure 2.20: RNA-Seq procedure [86].	39
Figure 3.1: Pre-processing of microarray data.	42
Figure 3.2: Gene expression matrix.	44
Figure 3.3: Chart of divisive hierarchical clustering scheme.	47
Figure 3.4: Linkage methods.	48
Figure 3.5: SOM neural network adapted from [108].	49
Figure 3.6: Flowchart of Bi-CoPaM adapted from [115]	51
Figure 3.7: UNCLES flowchart with type B of external specifications adapted from [118] ...	54
Figure 3.8: Support vector machine classifier.	56
Figure 3.9: A perceptron with m inputs and a bias.	59
Figure 3.10: MLP Artificial neural network.	62
Figure 3.11: Feature selection methods.	65
Figure 3.12: Train and test performance when changing the number of parameters in the classifier model adapted from [162].	66

Figure 3.13: Microarray data analysis.....	68
Figure 4.1: Immigration of a cuckoo towards goal habitat.	75
Figure 4.2: Flowchart of COA-GA.....	77
Figure 4.3: Proposed microarray data analysis procedure.	81
Figure 4.4: MLP vs SVM performance without clustering.....	86
Figure 4.5: Proposed shuffle method.	88
Figure 4.6: Accuracy and sensitivity of MLP classifier results for three cancer datasets when no clustering is used, compared to using the shuffle technique with COA-GA for clustering.	91
Figure 4.7: Accuracy and sensitivity of SVM classifier results for three cancer datasets when no clustering is used, compared to using the shuffle technique with COA-GA for clustering.	92
Figure 4.8: Cost minimisation for four algorithms over 100 iterations for leukaemia.	93
Figure 4.9: Cost minimisation for four algorithms over 100 iterations for lymphoma.	93
Figure 4.10: Cost minimisation for four algorithms over 100 iterations for prostate cancer.	94
Figure 5.1: Schematic of the general methodology for gene selection.	99
Figure 5.2: Frequency plots before (a) and after (b) discretisation for lymphoma dataset.	101
Figure 5.3: Frequency plots before (a) and after (b) discretisation for prostate dataset. ..	101
Figure 5.4: Frequency plots before (a) and after (b) discretisation for leukaemia dataset.	101
Figure 5.5: Analogy between musical improvisation process and optimisation process.	105
Figure 5.6: Flowchart of COA-HS.	108
Figure 5.7: Accuracy of SVM for selected genes by MRMR.	109
Figure 6.1: Junction reads [253].....	117
Figure 6.2: RNA-Seq alignment methods.....	117
Figure 6.3: Count table for RNA-Seq.....	118
Figure 6.4: Variance-mean dependence adapted from [260].....	123
Figure 6.5: Flattened exons' locations.	125
Figure 6.6: RNA-Seq data analysis.....	125
Figure 6.7: RNA-Seq analysis workflow.	127
Figure 6.8: Initial FASTQC output.	129
Figure 6.9: Per GC content for mutated sample after SortMeRna	131
Figure 6.10: Exons grouped by gene in GRangesList format.	133
Figure 6.11: RangedSummarizedExperiment format.....	134
Figure 6.12: Density of mean counts for each sample.	135
Figure 6.13: Probability of observing a given number of counts for all samples.	136
Figure 6.14: Natural scale for sample-sample visualisation.	137
Figure 6.15: Log2 normalised counts scale for sample-sample visualisation.	137

Figure 6.16: rlog scale for sample-sample visualisation.	138
Figure 6.17: PCA plot for all samples.	138
Figure 6.18: Dispersion estimates versus the mean normalised count from DESeq2.	139
Figure 6.19: Results of DESeq2.	140
Figure 6.20: MA plot of results using adjusted p-value > 0.1.	141
Figure 6.21: MA plot of results using adjusted p-value > 0.1 and log2 fold changes of at least double or half.	141
Figure 6.22: Heat map of top 25 differentially expressed genes.	142
Figure 6.23: R/Bioconductor session information for differential gene expression.	143
Figure 6.24: Count table in DEXSeqDataFrame.	145
Figure 6.25: Density of mean counts for all samples including group A (1-6) and B (7-12).	146
Figure 6.26: Probability of observing a given number of counts for all samples including group A (1-6) and B (7-12).	147
Figure 6.27: Dispersion estimates versus the mean normalised count from DEXSeq.	148
Figure 6.28: Results of DEXSeq.	149
Figure 6.29: MA plot for differential exon usage.	149
Figure 6.30: Mean expression level for exons of the FBgn0000382 gene.	150
Figure 6.31: R/Bioconductor session information for differential exon usage.	151
Figure 6.32: Log2 fold change of genes contributing to metabolic process of chitin.	153
Figure 6.33: Log 2 fold change of genes contributing to chitin-based cuticle development.	153
Figure 6.34: Schematic of the general methodology for RNA-Seq classification.	154
Figure 6.35: Accuracy of SVM classifier.	155

List of Tables

Table 2.1: Comparison of next-generation sequencing platforms.	29
Table 2.2: Summary of three quality score formats.....	37
Table 3.1 Activation functions	60
Table 4.1: Basic information of microarray data.....	82
Table 4.2: Number of genes in each cluster when data is clustered into two groups.....	82
Table 4.3: MLP classifier performances including the mean sensitivity, specificity, accuracy, and standard deviation (SD) for leukaemia.....	83
Table 4.4: MLP classifier performances including the mean sensitivity, specificity, accuracy, and standard deviation (SD) for prostate cancer.....	84
Table 4.5: SVM classifier performances including the mean sensitivity, specificity, accuracy, and standard deviation (SD) for leukaemia.....	85
Table 4.6: SVM classifier performances including the mean sensitivity, specificity, accuracy, and standard deviation (SD) for prostate cancer.....	85
Table 4.7: Basic information of the microarray data used in this study.	89
Table 4.8: The mean sensitivity (SE), specificity (SP), accuracy (AC), and standard deviation (SD) of classification results for MLP and SVM classifiers when integrating different clustering algorithms in the shuffle technique for the leukaemia dataset.....	89
Table 4.9: The mean sensitivity (SE), specificity (SP), accuracy (AC), and standard deviation (SD) of classification results for MLP and SVM classifiers when integrating different clustering algorithms in the shuffle technique for the lymphoma dataset cancer.....	90
Table 4.10: The mean sensitivity (SE), specificity (SP), accuracy (AC), and standard deviation (SD) of classification results for MLP and SVM classifiers when integrating different clustering algorithms in the shuffle technique for the prostate dataset cancer.....	90
Table 5.1: Basic information of the microarray data used in this study.	98
Table 5.2: Discretisation of gene expression data.	100
Table 5.3: Results of using HS with different PAR and HMCR values.....	107
Table 5.4: Means and standard deviations for the number of selected genes, sensitivity, specificity, and accuracy of SVM classifier for 20 runs of optimisation algorithms for prostate cancer dataset.	110
Table 5.5: Means and standard deviations for the number of selected genes, sensitivity, specificity, and accuracy of SVM classifier for 20 runs of optimisation algorithms for leukaemia cancer dataset.	110

Table 5.6: Means and standard deviations for the number of selected genes, sensitivity, specificity, and accuracy of SVM classifier for 20 runs of optimisation algorithms for Lymphoma cancer dataset.	111
Table 6.1: Library size affect.	119
Table 6.2: Summary of results for seven normalisation methods; 0 indicates not satisfactory, 1 indicates satisfactory, and 2 denotes very satisfactory (modified from [265]).....	120
Table 6.3: Estimated size factor for each sample using DESeq.	135
Table 6.4: Estimated size factors for each sample using DEXSeq.	145
Table 6.5: GO analysis.....	152

Acknowledgments

First, I would like to express my utmost appreciation and thanks to my first supervisor Dr Hasan Shaheed, for his invaluable guidance regarding the academic aspects of my research and for encouraging me during difficult times while working on my PhD. He has made the last four years of research at Queen Mary University an exciting experience. I would also like to specifically acknowledge my second supervisor, Professor Hazel Screen, who always gave me detailed constructive feedback and encouraged me in my academic pursuits.

I also thank Professor Marta Korbonits and Dr Sayka Barry, who are based at the Barts and London School of Medicine, and who provided me with the RNA-Seq and microarray datasets.

My profound gratitude goes to my parents, who have always been supportive and have made tremendous sacrifices to make sure I get the best education. I will always remember all their acts of kindness and would like to dedicate this thesis to them.

Chapter 1: Introduction

1.1: Background

A basic question in regulatory biology is 'how is it possible that all humans can have a 99.9% identical genome, but still be different?' Each individual has a 100% identical genome in different organs, but each organ has a different shape and functionality. The answer is that different cells types express a different set of genes through a phenomenon known as gene expression, in which the information from the DNA is transcribed to RNA, and then translated to proteins that form the cell shapes and their functionalities.

Many studies have shown that if the sequence of a gene is mutated, it can result in an unscheduled production of protein, which can lead to diseases such as cancer [1]. Furthermore, recent research suggested that cancer could also form without any change happening in the underlying gene sequence itself, through epigenetic modifications such as histone modification and DNA methylation [2]. The invention of microarray technology paved the way to quantify the gene expressions of thousands of genes simultaneously. More recently, a more sophisticated technology known as Next Generation Sequencing (NGS) has improved gene expression quantification and allowed investigation of epigenetic modifications on a wide scale in genomes.

Microarray technology can be used in a range of scientific fields and can contribute to the diagnoses of diseases. Although cancer is a complex disease that arises from multiple genetic factors, it is known that the level of gene expression could carry a signature for a disease. A vast majority of fatal diseases have a unique gene expression profile that can be observed using microarray technology. The main field that microarray gene expression is applied in is profiling cancerous tissue. Measuring the gene expression of diseased tissue enables researchers to understand tumours and discover possible markers for them. For example,

some prognostic markers were discovered based on gene expression profiles by Sorlie *et al* [3] which are used for overall survival in breast cancer.

Analyses of gene expression data produced by microarray technology can be classified into two different types; namely supervised and unsupervised learning (clustering). Clustering divides data sets into several groups such that the similarity within a group is greater than that among groups. Since copious amounts of gene expression data is produced with microarrays, it is useful to group genes such that genes with similar expression patterns are put into one cluster, where the genes within the cluster are known as co-expressed genes. Research suggests that genes in one cluster have related functions [4–6]. In machine learning, procedures that use annotated samples are referred to as supervised learning [7]. Therefore, in supervised learning, classes are predefined and the objective is to train a set of data to form a classifier for classification of future observations.

Recently, microarray technology has been used to determine subtypes of certain cancers based on differences in the expression level of key genes [8–10]. This approach has become known as cancer classification, and provides detailed information on the genetic makeup of each individual cancer patient, thereby potentially improving the accuracy of treatment decisions made by doctors [11]. During microarray analysis, the number of genes is significantly higher than the number of samples [12,13] and classification with a high degree of accuracy is challenging, due to the phenomenon of the so called curse of dimensionality [14,15]. In order to overcome this problem, gene selection mechanisms have been introduced, by which only the most important genes are selected and used for classification purposes [16–19]. There are several advantages to this process of minimising the number of genes and only selecting the meaningful genes that are more predictive during classification, and this will be explained in detail in chapter 3.

Whilst microarray technology is widely used and has greatly contributed in gene expression research over the years, it does have its limitations, such as the noise produced during the experiments [20]. Therefore, over the last few years, new sequencing technologies have been developed including next generation sequencing (NGS) technology. The arrival of NGS technologies in the marketplace has changed the scientific perspective on basic, applied and clinical research. NGS technologies have the ability to produce millions of sequence reads in each run, which makes it possible to sequence the whole genome easily. As a result, it allows large-scale evolutionary and comparative studies to be performed. NGS technologies have been used in different projects, such as RNA expression profiling, mutation discovery, defining DNA-protein interaction, and whole-genome sequencing [20].

RNA expression profiling, which utilises NGS technology, can provide gene expression quantification on a genome-wide scale, providing a tool to not only discover differentially expressed genes between conditions, but also one that enables researchers to investigate differentially expressed isoforms, and differential exon usage across different conditions. Furthermore, RNA-Seq data can be used to discover novel transcripts using statistical analysis [21].

1.2: Motivations

Over the last 20 years, there has been a revolution in biological sciences and technologies like microarray and NGS, which provide an overwhelming volume of data that requires computational tools to sift through this data to provide useful information for more informed treatment decisions. Most types of cancers are treatable if they can be detected at an early stage. The determination of cancer type and stage is also crucial when choosing an appropriate treatment. Therefore, the development of computational tools is an important topic, and more research is needed to make the most out of the available data.

Although several methods have been proposed to increase the accuracy of classification [22,23], more research is still needed to propose new models that can further increase the prognosis and classification accuracy of diseases such as cancer. Achieving high classification accuracy is of the utmost importance for personalised medicine, as it would lead to more informed decisions by doctors and subsequently save patients' lives. Since the invention of NGS and RNA-Seq, there have been numerous pipelines to investigate the analysis of such data. Nevertheless, due to the rapid development of statistical methods for RNA-Seq, the proposed pipelines have undergone several changes to improve the results. Hence, it is essential to provide a state-of-the-art pipeline to enhance scientific discoveries and their implementation into clinical practice.

1.3: Aim and Objectives

This thesis presents an investigation into the analysis of gene expression using microarray and next generation sequencing data for multi-category diseases like cancer, in order to create useful information towards personalised medicine. The objectives of this thesis are described below:

- Explore different methods for analysing gene expression data
- Incorporate machine learning techniques to cluster and classify gene expression data

- Assess optimisation-based algorithms for data clustering to enhance the accuracy of cancer classification
- Investigate the impact of data clustering prior to gene selection on classification accuracy
- Develop novel gene selection models to select the highly informative genes using microarray data
- Identify differential gene expression and differential exon usage using RNA-Seq

1.4: Contributions

In this study, different approaches are adopted to improve prognosis and classification of multi-category diseases such as cancer using microarray and NGS data towards more personalised medicine. The main contributions of this investigation are as follows:

- An innovative gene selection approach using the shuffle method prior to cancer classification is proposed. It is noted that in cancer classification using clustering based gene selection, changing the number of clusters results in the selection of different sets of genes due to the random initialisation of centroid protocol for clustering methods. This results in a variation in the accuracy of classification. In order to overcome this problem, the shuffle method is proposed, in which genes with a higher rate of repetition are selected after six runs of the clustering algorithm.
- A novel optimisation algorithm, COA-GA, has been developed by integrating the Cuckoo Optimisation Algorithm (COA) [24] and the Genetic Algorithm (GA) [25] to enhance classification performance. The proposed algorithm (COA-GA) not only outperforms COA, GA and Particle Swarm Optimisation (PSO) at achieving a better classification performance, but also reaches a better minimum with only few iterations.
- It is additionally confirmed that traditional clustering does not have any impact on gene selection and classification performance. However, optimisation based clustering is shown to enhance the accuracy of gene selection and classification.
- The performances of two well-known classification methods, SVM and MLP, are assessed. To examine the performance of these two methods, different cancer datasets including leukaemia, prostate, and lymphoma were utilised in different setups. Higher classification

accuracy is observed in all cases, with the SVM classifier being compared to MLP when analysing gene expression datasets.

- A novel optimisation algorithm, COA-HS, has been developed to enhance gene selection. This optimisation algorithm was then used in a two-stage gene selection method, MRMR-COA-HS, in order to select a few genes that could provide high accuracy in cancer classification. Comparative performance assessment of the proposed method with other evolutionary algorithms, suggest that the proposed algorithm significantly outperforms other methods in selecting a lower number of genes, while maintaining the highest classification accuracy. The functions of all selected genes using this method were investigated further, and it was confirmed that the selected genes are biologically relevant to each type of cancer.
- A state-of-the-art pipeline for RNA-Seq data is proposed. This pipeline was used to analyse a set of RNA-Seq data, which was produced at the Genome Centre of Queen Mary University. In the proposed pipeline, differential gene expression, and differential exon usage were investigated in detail. Furthermore, functional genomics of differentially expressed genes were investigated, and some key genes were identified for the case study under investigation that can be used as biomarkers. Finally, the application of data classification for RNA-Seq data was explored, and methods similar to those in microarray classification were successfully implemented.

1.5: Outline of the Thesis

Chapter 2 presents a review of recent developments. Initially, a brief overview of the biological aspects of the thesis such as gene expression phenomena is given. Then, it explores different types of microarray technologies like cDNA and oligonucleotide microarray. Different NGS technologies are then investigated, and a unique method for each technique is explained. At the end of the chapter, the techniques that incorporate NGS technology such as RNA-Seq are assessed.

Chapter 3 describes the main steps required for microarray analysis, including pre-processing, clustering, and classification. In this chapter, several clustering methods such as K-means, C-means, Hierarchical clustering, self-organising map, Bi-CoPaM and UNCLES are investigated. Then the most widely used classification methods such as SVM and MLP artificial

neural networks are briefly explained. Afterwards, the importance of gene selection before classification is investigated

Chapter 4 concerns the effect of gene clustering prior to gene selection on the classification performance. Initially, the effects of traditional data clustering methods on the classification performance are investigated. Then, the effect of optimization based clustering algorithms on the performance of SVM and MLP classifiers is investigated and compared to conventional methods.

Chapter 5 presents the development of a two-stage gene selection process, using MRMR and the COA-HS algorithm. In this chapter, the MRMR method is described first. Then the use of optimization algorithms for gene selection as well as the proposed objective function are explained. To this end, different optimization algorithms such as GA, PSO, COA, and HS are investigated. The use of the Leave-One-Out Cross-Validation (LOOCV) method to evaluate the performance of our proposed method is then explained.

Chapter 6 is divided into two main sections. The first section describes the main steps required for RNA-Seq analysis, including experimental considerations in design, pre-processing and quality assessment, alignment, building a count table, and normalisation. Then the concept of differential expression at the gene and transcripts levels are examined, and some of the well-known software for such analyses are identified. In the second section of this chapter a state-of-the-art pipeline for RNA-Seq analysis is presented. In this chapter, RNA-Seq data from AIP deficient *Drosophila* is used as the case study (6 samples). Initially, different pre-processing steps are explained in order to eliminate biological and technical noises that present in RNA-Seq data. Approaches used to create a count table after mapping the samples to a reference genome are then explained. Finally, downstream analysis and classification are explored.

Chapter 7 consists of conclusions, discussion, and future work.

1.6: Publications

Publications extracted from this thesis are outlined below.

- Elyasigomari, V., Mirjafari, M. J., Screen, H R C., and Shaheed, M H. (2015), Cancer classification using a novel gene selection approach by means of shuffling based on data clustering with optimization, *Journal of Applied Soft Computing*. 35(1), pp 43-51.
- Elyasigomari, V., Lee, D., Shaheed, M H. (2017), Development of a two-stage gene selection method that incorporates a novel hybrid approach using the cuckoo optimization

algorithm and harmony search for cancer classification. *Journal of Biomedical Informatics*. 67(1), pp 11-20.

1.7: Utilised microarray datasets throughout this thesis

In respect to cancer gene expression studies, there are several benchmark microarray data sets including leukaemia, lymphoma, and prostate cancer data sets which are also used in this research. Brief explanation on these datasets are given in following subsections.

1.7.1: Leukaemia dataset

This dataset was taken from a collection of leukaemia samples by Golub *et al.*, (1999). In total 72 patients participated in their study. As a result, 72 samples were obtained from bone marrow (63 samples) or peripheral blood (9 samples) of these patients and the gene expression for these samples were measured using Affymetrix high-density oligonucleotide arrays (Affymetrix Hgu6800 chips) that contained 7129 genes. From 72 patients, 47 were associated with acute lymphoblastic leukaemia (ALL) and 25 were diagnosed with acute myeloblastic leukaemia (AML) [26].

Although the original study was designed for leukaemia classification in the case of two class classification, the 47 samples from ALL could be further categorised into ALL B-CELL (38 samples), ALL T-CELL (9 samples) which made it possible for some studies to use this dataset for multiclass classification. It is noted that this dataset has been used by many authors to test the accuracy of their techniques. Golub and his colleagues normalised this dataset such that overall intensities for each chip became equivalent by re-scaling intensity values [27]. The original dataset available from the Broad institute can be accessed using the link: http://portals.broadinstitute.org/cgi-bin/cancer/publications/pub_paper.cgi?mode=view&paper_id=43

1.7.2: Lymphoma dataset

Diffuse large B-cell lymphoma (DLBCL) is an aggressive malignancy of mature B lymphocytes. The DLBCL dataset provided by Alizadeh *et al.*, (2000) consists of 47 samples and each sample contains 4,026 genes. 24 samples were obtained from germinal centre B-like DLBCL, and the remaining 23 samples were acquired from activated B-like DLBCL. For the measurement of gene expression levels, specialised cDNA microarray was used which consisted of genes that had known to have immunologic/oncologic importance or had expressed in lymphoid cells [28].

In the process of hybridization, a tumour mRNA sample was used for the fluorescent cDNA targets (labelled with dye Cy5) and for the fluorescent cDNA reference a mRNA sample was used from lymphoma cell lines and labelled by Cy3. Then the GenePix 4000 microarray scanner was used to obtain the fluorescent images. In order to calibrate the fluorescent ratios for all arrays a single scaling factor was calculated such that on each array the median fluorescence ratio of well-measured spots was 1.0 [28]. This scaling factor was applied to all fluorescence ratio for each array. It is noted that fluorescent intensities above 1.4 times of background was considered as well measured. The fluorescent ratios then were log-transformed (base 2). In order to eliminate the effect of the amount of RNA in the reference pool each data point was centred by subtracting the median value for all genes [29]. The original dataset available from Lymphoma/Leukemia Molecular Profiling Project can be accessed using following link <https://llmpp.nih.gov/lymphoma/index.shtml>.

1.7.3: Prostate dataset

Prostate cancer is one the most common heterogeneous cancer among humans. Singh *et al.*, (2002) used microarray expression analysis to investigate important genes and pathological features that underlie global biological differences in prostate cancer. In their experiment, total RNA was isolated from 55 samples that were obtained from patient with prostate cancer and 53 samples that were acquired from healthy individuals. These samples were labelled by biotin and hybridized to HU95Av2 microarrays that contained 12,600 genes and expressed sequence tags [30].

Affymetrix GeneChip software was used to calculate the average differences. The average pixel mean and standard deviation values for each probe set and the standard deviation of the average difference for all genes were calculated. Based on these calculations some samples which had high standard deviation were removed from the experiment and 102 samples including 50 healthy and 52 cancerous samples were chosen as high quality samples. These samples then were scaled to a reference sample. Afterwards, the relative variation of expression for each gene was computed using the minimum and maximum expression values of the gene across all samples. The original dataset available from the Broad institute can be accessed using following link:

http://portals.broadinstitute.org/cgi-bin/cancer/publications/pub_paper.cgi?mode=view&paper_id=75

Chapter 2: Background and Recent Developments

All living organisms are composed of cells. All cells are characterised by a plasma membrane, which encapsulates the cytoplasm and provides internal space where important functions are carried out (see Figure 2.1). In this internal compartment, different components are present, such as ribosomes and the nucleus. Ribosomes are organelles that process the cell's genetic information to create protein. In the nucleus, the heredity information is stored in the form of Deoxyribonucleic Acid (DNA). Most DNA molecules are double-stranded helices consisting of four different nucleotides which are: guanine (G), adenine (A), thymine (T) and cytosine (C) [31].

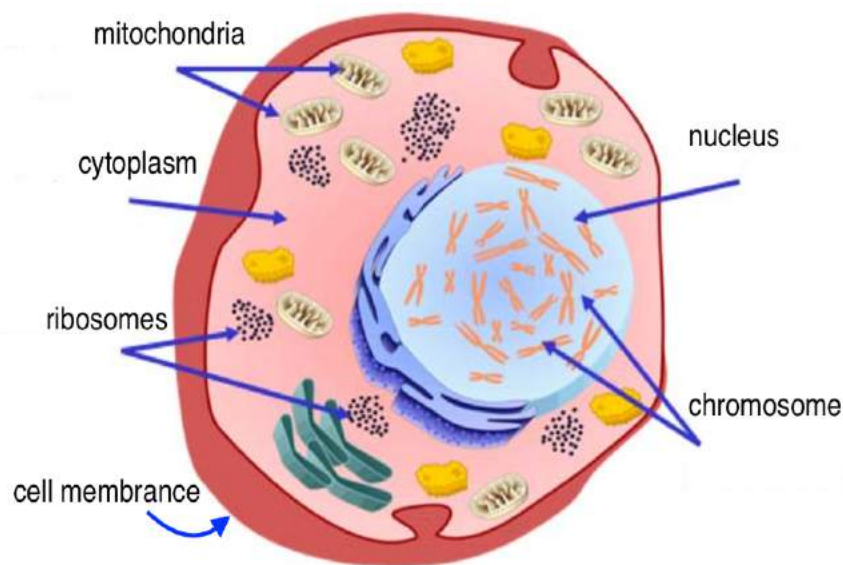


Figure 2.1: Cell structure [32].

In humans, the complete set of genetic information that is required for normal functioning of the body consists of 3 billion base pairs of DNA packaged into 23 chromosomes. Each cell contains almost 2 meters of DNA, and each human roughly consists of 50 trillion cells. If all

of the DNA was uncoiled, it would wrap around the Earth's equator 2.5 million times [33]. So the question is: how is this incredible amount of DNA stored in a nucleus?

The answer to this question lies in the fact that certain proteins compact chromosomal DNA into the microscopic space of the eukaryotic nucleus. These proteins are called histones, and the resulting DNA-protein complex is called chromatin.

As can be seen in Figure 2.2 point A, at the simplest level chromatin is a double-stranded helical structure of DNA. At point B, DNA is wrapped around eight histones 1.65 times to form a nucleosome. Histones are positively charged proteins named H1, H2A, H2B, H3, and H4 [34]. Since DNA is negatively charged, histones bind with DNA very tightly. A nucleosome with the H1 histone is known as a chromatosome (point C). At point D, it is illustrated how nucleosomes fold up to form a 30-nm fibre. This 30-nm fibre folds up more to form loops averaging 300-nm in length. Afterwards, the 300-nm fibres are compressed and folded to produce a fibre that is 250-nm in width and 700-nm in length (point E). Finally, tight coiling of the 250-nm fibre produces the chromatid of a chromosome [33].

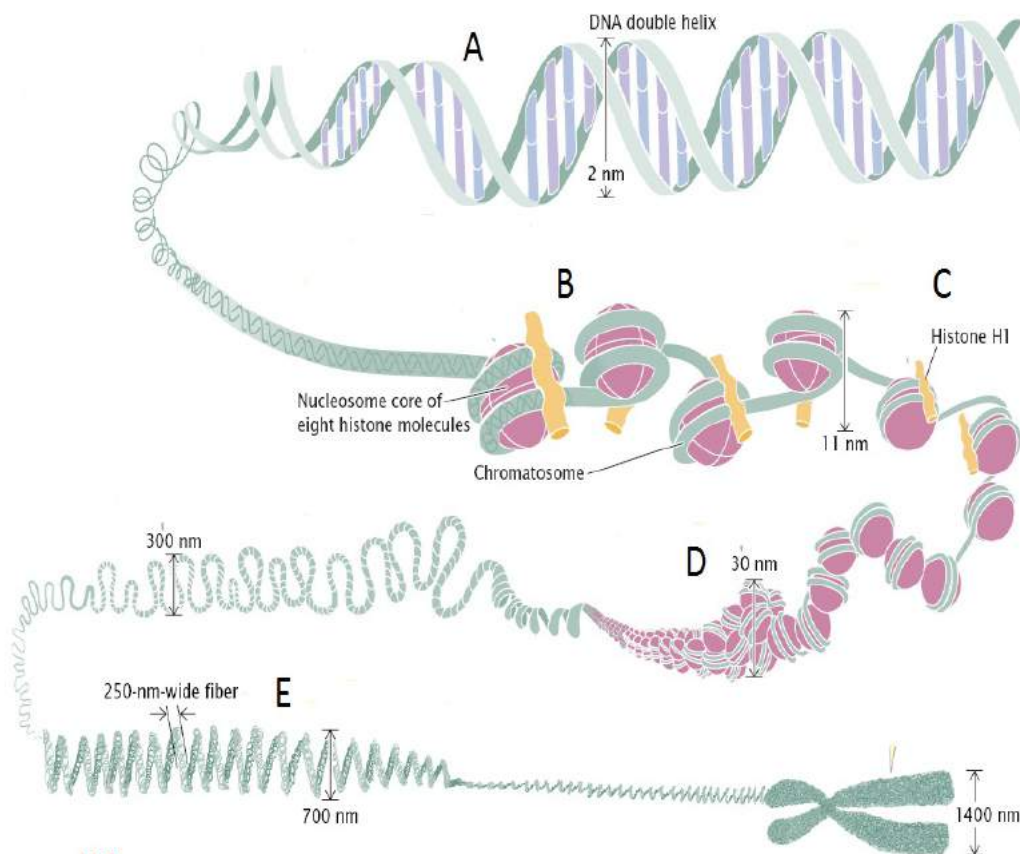


Figure 2.2: Chromatin structure adapted from [33].

2.1: Gene Expression

Gene expression refers to all the processes that convert genetic information from the DNA sequence of genes into gene products. These products can range from proteins to functional RNAs that result from protein and non-protein coding genes respectively. A gene is a segment of DNA that consists of information used to code for a protein. The genetic information that codes for the production of amino acids is stored as three-letter codes, called codons, and the sequence of codons defines the primary structure of the final proteins [35].

Gene expression involves two steps: the first step is “transcription”, which refers to the synthesis of a ribonucleic acid (RNA) molecule using DNA, which occurs within the cell nucleus. In this step, the transcription factor connects to the part of DNA referred to as the TATA box (also called the Goldberg-Hogness box). Afterwards, the RNA polymerase binds to the transcription factor, thereby adding energy (adenosine triphosphate (ATP)) to the process. At this point, the transcription starts, and then finally the process is terminated by the RNA polymerase, and the newly formed RNA is released from the DNA (see Figure 2.3). Then it travels in the form of messenger RNA (mRNA) to the edge of the nucleus, where it gains access to the cytoplasm through a tiny hole called a nuclear pore [36].

The second step is “translation”, which is carried out in the cytoplasm. This step refers to the process of facilitating the codon within the mRNA for the synthesis of a special protein. In this step, two important components are utilised: the ribosome (rRNA) and transfer RNA (tRNA). After mRNA is exported to the cytoplasm, it is attached to the ribosome. Amino acids are carried by tRNA, and can only be added to the chain of growing protein if the tRNA aligns to its complementary mRNA codon. Therefore, as the name suggests, the genetic information translates into chain of proteins [36].

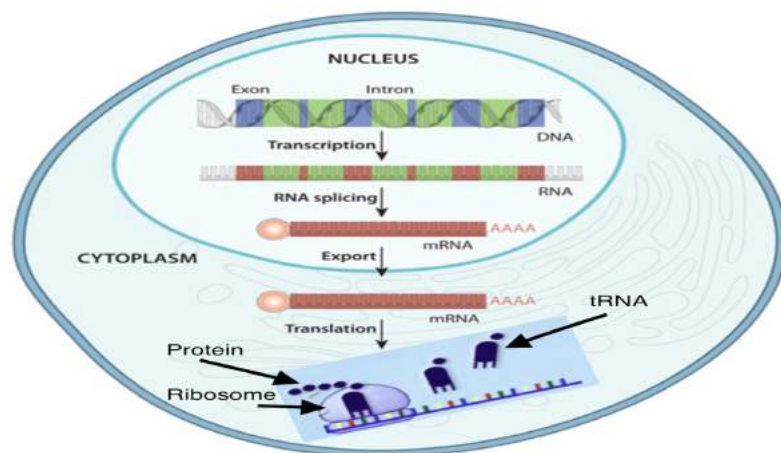


Figure 2.3: Process of transcription and translation adapted from [37].

2.2: DNA Microarray

Microarray technology is a powerful way to quantify gene expression. By using microarray, it is possible to examine the expression level of thousands of genes in one experiment. It can be used to compare the expression of many genes under different conditions, such as cancerous cells versus normal cells. Although there are several microarray technologies that exist to date like exon arrays [38], high resolution tiling arrays [39], and Illumina bead arrays [40], two technologies are specifically used in practice, cDNA and oligonucleotide microarray [41].

2.2.1: cDNA Microarrays

In the case of cDNA microarrays, the production of arrays begins with the selection of total or partial fragments of cDNA to be printed on the array. Partial fragments of cDNA are known as expressed sequence tags (ESTs). cDNA clones are usually selected from available databases, including Unigene [42], dbEST [43], and GeneBank [44]. The chosen cDNA clones are then amplified using polymerase chain reaction (PCR), and purified before using high-speed robots to print them on a coated glass surface. These immobilised cDNA clones on the glass are known as microarray probes, and each probe represents a specific gene (see Figure 2.4).

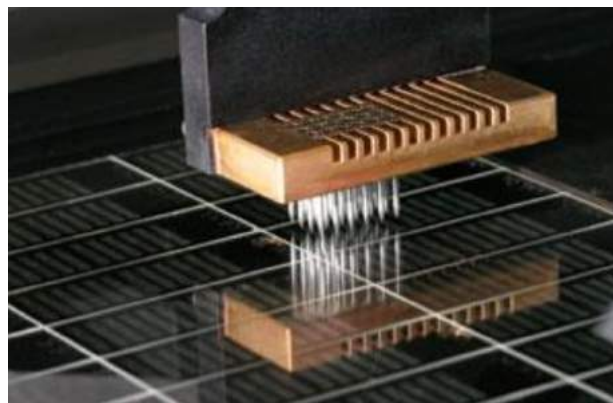


Figure 2.4: Microarray glass [45]

When comparing the gene expressions of two samples, the first step is extracting RNA from the cells and amplifying it using a polymerase chain reaction (PCR). After the PCR products are cleaned, they are reverse transcribed into cDNA by using an enzyme reverse transcriptase and nucleotides labelled with different fluorescent dyes by chemically attaching the dye molecules to the ends of the corresponding cDNA strands [46]. For example, cDNA from cells grown in cancerous conditions are labelled with a red dye (Cy5) and cDNA from cells grown

in healthy conditions are labelled with a green dye (Cy3). Once the samples have been labelled with different fluorescent dyes known as probes, they can be hybridised onto the same glass slide of the array, where any cDNA sequence attaches to its complementary sequence on the glass. Unattached materials are gently removed and the glass is left to dry. The spots are then excited by a laser and scanned afterwards. Specifically, two scans are required for each microarray. The first scan is for the red fluorescent and the second scan is to detect the green fluorescent. After these two scans, a three-colour image is typically composed, containing red, yellow, and green spots, which refers to highly expressed, equal expressed and less expressed genes correspondingly [41]. It is shown in Figure 2.5 that some spots are shown in black, which can be explained by the fact that none of the samples contain significant amounts of the corresponding type of RNA, or a mistake in the hybridisation process.

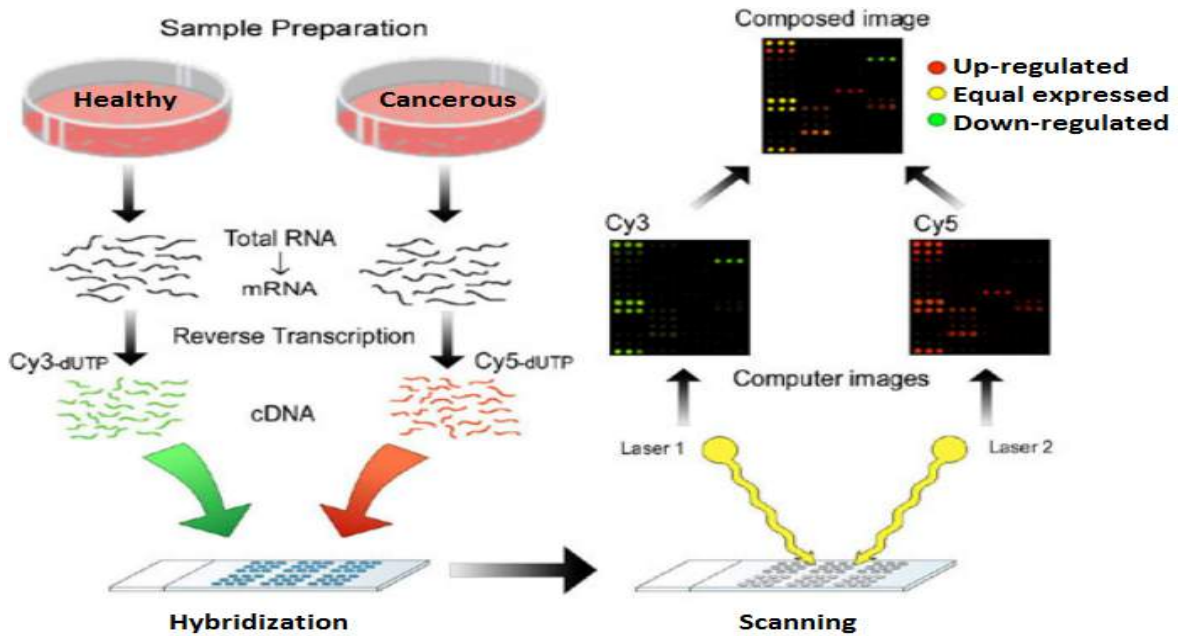


Figure 2.5: DNA microarrays technology modified from [47].

Once the image is generated, it is analysed to identify the spots using special software, where the background hybridisation can be estimated and the intensity is calculated for each spot. Afterwards, the expression ratio is calculated as a primary comparison tool to relate the intensity of red and green lights as below:

$$T_k = \frac{R_k}{G_k} \quad (2.1)$$

where T_k , R_k and G_k are the expression ratio, the intensity of the sample and the intensity of the reference (healthy) samples respectively.

2.2.2: Oligonucleotide Microarray

In contrast to the cDNA microarray that uses the complete gene sequence as targets, oligonucleotide microarrays (single-channel) use a number of short oligonucleotide sequences (usually 20-70 nucleotides long) that represent a specific gene. These oligonucleotide sequences can be either spotted or synthesised on the array surface. Although there are varieties of oligonucleotide arrays, Affymetrix GenChip is the most widely used technique [48].

In Affymetrix GenChip, a short stretch of oligonucleotide strands is used, and the spots are synthesised through photolithography. The fabrication of the array is based on the sequential addition of nucleotides to the microarray surface (wafer), which is chemically protected from nucleotide additions until exposure to UV light. Photolithographic masks are used to place nucleotides on specific probe sites, and the sequential addition of lithographic masks determines the order of sequential synthesis on the array (Figure 2.6). In this method, each gene is represented by 25 pairs (25-mer) of oligonucleotide [49].

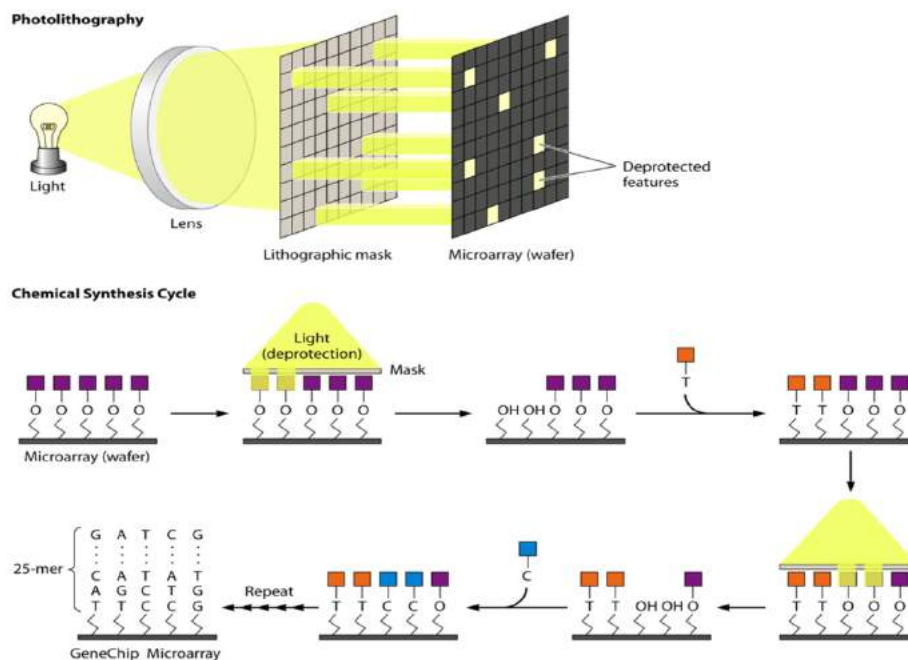


Figure 2.6: Affymetrix GenChip lithography [50].

One strand of the 25-mer sequence is known as a perfect match (PM) and the other is referred to as a mismatch. The sequence of the nucleotide in the perfect match probe is exactly complementary to a particular gene, and thus measures the expression of the gene. However, the mismatch probe differs from the perfect match probe by a single base nucleotide at the centre position of the probe, which prevents the target from binding to the gene transcript (see Figure 2.7). The main reason for using the mismatch probe is to determine the

background and nonspecific hybridisation that contributes to the measured signal for the perfect match oligonucleotide probe [36].

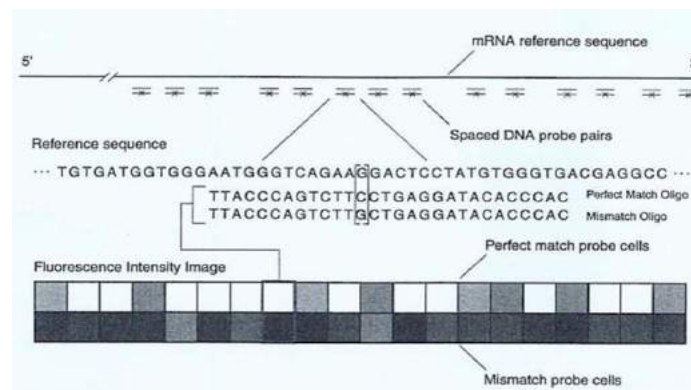


Figure 2.7: Affymetrix expression array design adapted from [51].

In the process of achieving expression, RNA is extracted from the sample and after amplification, is labelled with chemical biotin. Then the labelled RNA is added to the Affymetrix array to bind with the relevant oligonucleotide probe. After washing the unattached materials, fluorescent stain that is capable of attaching to the biotin on the RNA is added to the array. The array is then scanned to obtain an image. In contrast to cDNA, Affymetrix arrays use a single dye colour (one channel), and therefore each sample should be added to a separate array and cannot be hybridised (see Figure 2.8) [41]. The hybridisation intensity of the perfect match and the mismatch is computed and subtracted from each probe by Microarray Suite software. As a result, the absolute intensity value for each probe is acquired. Afterwards the intensity of the perfect match is subtracted from the intensity of the mismatch probe. Therefore, the average intensity for each gene can be calculated. Then the intensity is converted into a ratio. Eventually, as in cDNA, the data is stored in the form of the gene expression ratio [52].

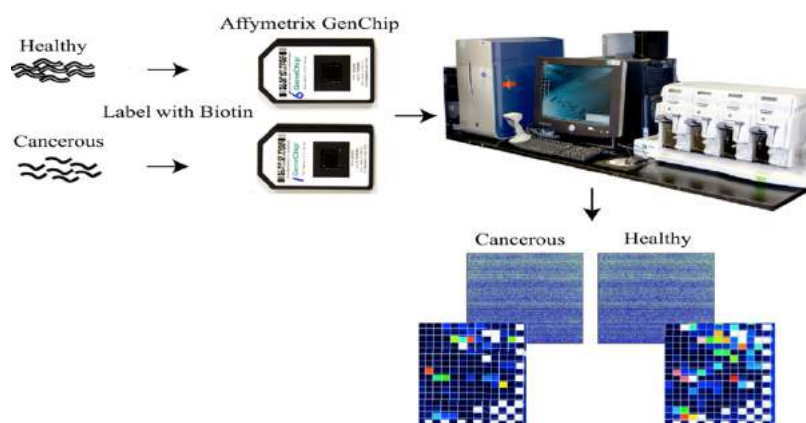


Figure 2.8: Affymetrix GenChip microarray modified from [53].

2.3: Next Generation Sequencing Technology (NGS)

DNA sequencing refers to the precise identification of the order of nucleotides (A, G, T and C) in a DNA sample. Over the last 4 decades, there have been major advancements in DNA sequencing technologies. In 1975, DNA sequencing by primed synthesis with DNA polymerase was investigated by Sanger and Coulson [54], and this method was then improved by utilising chain-terminating inhibitors [55]. The automated readout of the sequence was successful when fluorescent tags were added to the chain terminator [56]. This innovation is known as First Generation Sequencing technology or Automated Sanger Sequencing. Over the last decade, technological advancements in the field of sequencing have introduced Second Generation Sequencing also known as Next Generation Sequencing (NGS) [57]. Not only has the cost of sequencing with the new methods been significantly reduced, but this technology is also capable of producing a significant amount of data in a shorter time [20,58].

Although there are several next generation sequencing platforms available, three platforms are more widely used: Roche 454, Illumina, and SOLiD [59]. The amount of reads and the length of each read (base pair) are varied across different platforms [60], but each platform has their own advantages and disadvantages [58], which are listed in Table 2.1.

Table 2.1: Comparison of next-generation sequencing platforms.

Technology	Read length	BP per run	Advantages	Disadvantages
ILLUMINA	30–40 (bp)	1 Gb	The most commonly used platform.	Low multiplexing capability of samples
ROCHE 454	200–300 (bp)	80–120 Mb	Fast run times; Longer reads	High cost; high error rates in homo-polymer repeats
SOLID	35 (bp)	1–3 Gb	Inherent error correction by two-base encoding	Long run time

All NGS platforms detect the order of nucleotides through 3 primary steps. The first step is the random fragmentation of DNA and ligation with some custom adaptors, known as sample preparation. Then the amplification step follows in which the fragments are amplified to produce a detectable signal. The last step is to perform sequencing reaction and detection of nucleotides in a sequential order one by one. In the following sections, the details of Roche

454 FLX Pyrosequencer, Illumina genome analyser, and SOLiD Applied Biosystems are analysed.

2.3.1: Roche/454 FLX Pyrosequencer

In this method, DNA is converted into sequence data through three main steps which are: DNA sample preparation, loading DNA samples onto beads, and finally sequencing DNA with the Genome Sequencer FLX instrument. Sample preparation starts with random fragmentation of DNA (400-600 bp), then the adaptors are attached to these fragments. Finally, the double-stranded DNA fragments are separated into single strands (see Figure 2.9).

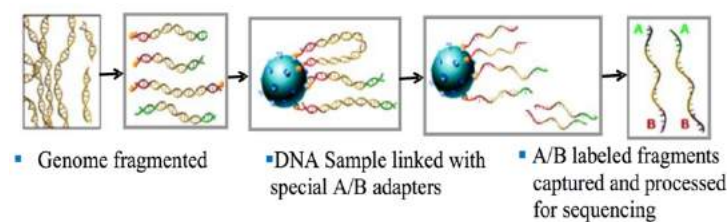


Figure 2.9: Sample preparation in Roche/454 [61].

In the second step, the fragments that are attached to adaptors are put onto micron-sized beads, which have a complimentary sequence to the adapter, then through using emulsion-based PCR, around ten million copies of each DNA fragment that is immobilised on the capture beads are produced (see Figure 2.10). Emulsion refers to a method where a single DNA molecule is isolated in aqueous micro-reactors by utilising water and oil emulsion. This amplification is required to generate sufficient signals that are detectable in the sequencing step [20].

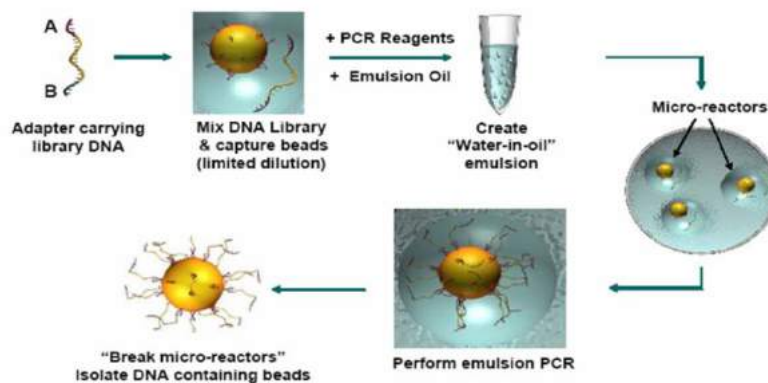


Figure 2.10: Amplification step in Roche/454 [61].

The last step is sequencing through synthesis. For this step, the beads are put into a well on a PicoTiter Plate along with an enzyme that helps the sequencing reaction (see Figure 2.11).

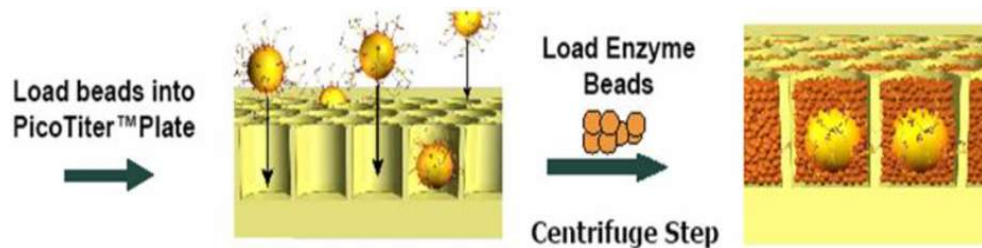


Figure 2.11: Sequencing by synthesis step in Roche/454 [61].

To accomplish this objective, starting from one end of the single-stranded fragment and based on the order of nucleotides in the strand, the enzyme synthesises the complimentary fragment through the sequential adding of nucleotides. Each time a nucleotide is added, a light is emitted which is then recorded by a camera (see Figure 2.12).



Figure 2.12: Sequencing by synthesis step in Roche/454 [62].

2.3.2: Illumina Genome Analyser

The Illumina Genome analyser converts DNA into sequenced data through three steps: sample preparation, cluster generation, and sequencing. After the random fragmentation of DNA, the ends of these fragments need to be repaired by adding the complementary nucleotides to the appropriate end of the fragment. Afterwards the ends of the fragment are phosphorylated, and a single A base is added to its 3' ends. Finally the adapters are ligated (see Figure 2.13) [63].

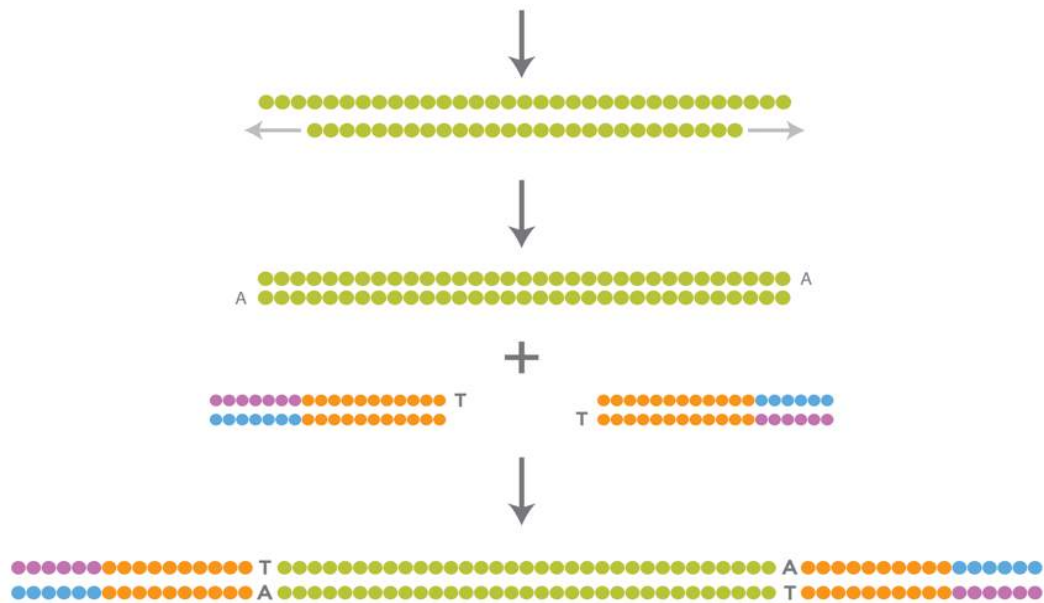


Figure 2.13: Sample preparation for Illumina sequencer [64].

In the second step, the prepared fragments are attached to a flow cell on a solid surface that has their complementary adapters. Once they are attached, the other side of the fragment also attaches to the solid surface and forms a bridge. Then a replica of the fragments forms, and they detach from each other. This cycle, known as bridge amplification, continues until clusters are formed [20].

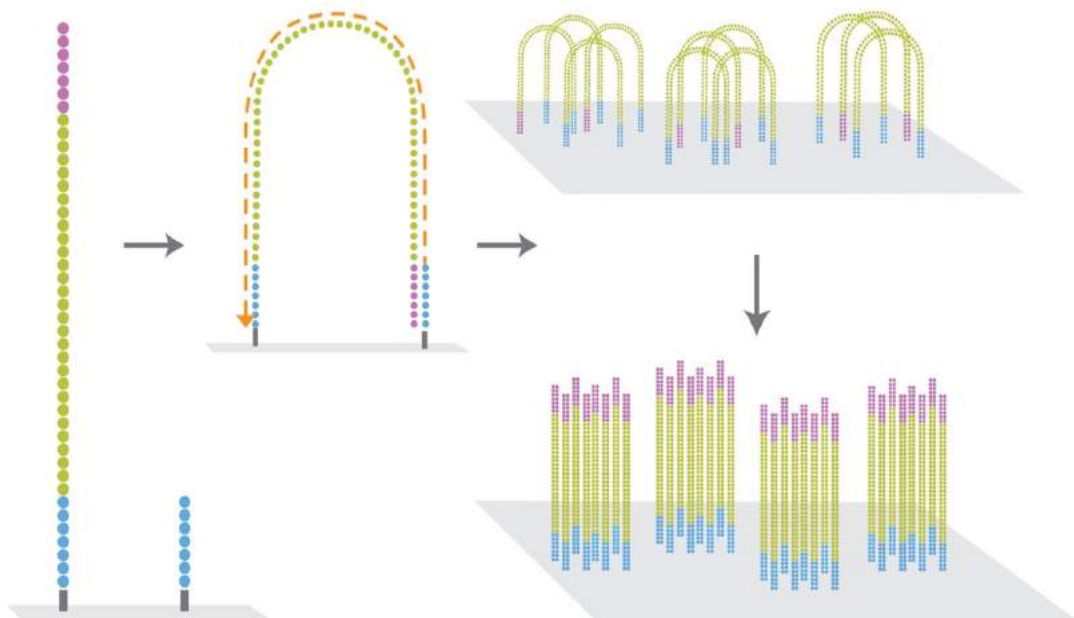


Figure 2.14: Bridge amplification for Illumina sequencer [64].

The final step is the detection of nucleotide sequences through sequence by synthesis. For this reason, DNA polymerase is added to the clusters on the slide and flooded by nucleotides. These nucleotides are engineered to have different colours corresponding to the base, and modified in such a way that the polymerase can extend by one base at a time through the use of a terminator. Once a nucleotide is attached to the fragment, the remaining unattached nucleotides are removed and a camera detects which nucleotide (C, A, G or T) is attached. Then the terminator is removed, and the slides are flooded by nucleotides again, and this process is repeated until the whole fragment is sequenced (see Figure 2.15). Although the sequencing and detection of nucleotides is done for one base at a time for each fragment, it is done for millions of fragments at the same time (see Figure 2.16) [64].

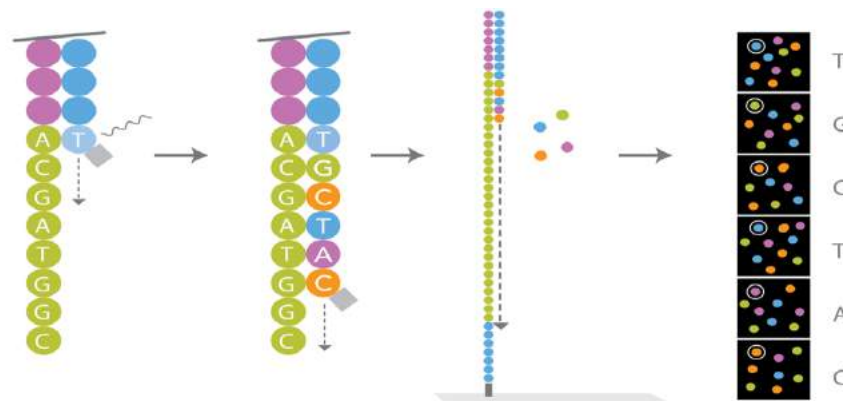


Figure 2.15: Sequencing by synthesis step in Illumina sequencer [64].

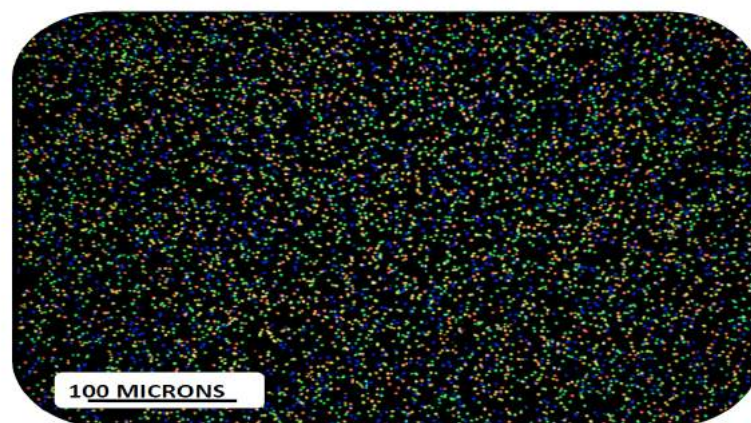
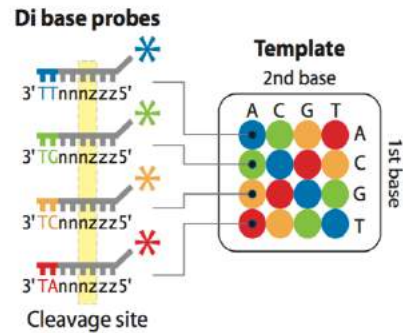


Figure 2.16: Pseudo colour enhanced image [65].

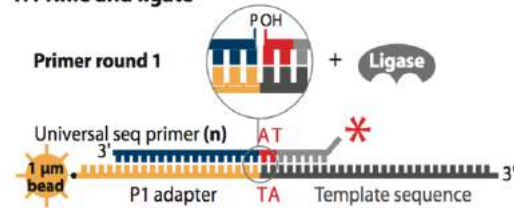
2.3.3: Applied Biosystems SOLiD Sequencer

The process of detecting the order of nucleotides by SOLiD (Sequencing by Oligonucleotide Ligation and Detection) technology is carried out through three steps: sample preparation, amplification and sequence by ligation. The sample preparation step is similar to other NGS platforms and consists of DNA fragments that are ligated to oligonucleotide adapters [66]. These ligated fragments are then amplified using emulsion based PCR as explained in Section 2.3.1 for Roche 454.

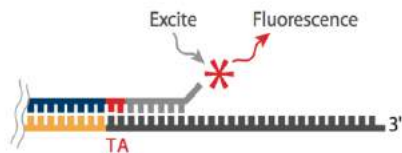
The sequencing by synthesis step in SOLiD technology is different from other NGS platforms, as this is done by DNA ligase (see Figure 2.17) as opposed to using a polymerase [64,67]. Initially a sequencing primer is hybridised to the P1 adaptor, which is attached to the bead. A mixture of Di-base probes that is labelled with four different fluorescent dyes races to ligate to the sequencing primer. After ligation, the fluorescent dyes are excited, and subsequently an image is taken. Afterwards, unextended strands are capped and fluorophores are cleaved. A new cycle starts 5 bases away from the priming site, by attaching another Di-base probe, and this is repeated for 7 cycles. Then the first sequencing primer is detached, and a new primer is attached to the template sequence (reset), and another 7 cycles is repeated for the new primer. In total, 5 rounds of this primer reset is performed (n, n-1, n-2, n-3, and n-4). As can be seen in Figure 2.17, eventually 35 bases are sequenced twice, thus improving sequencing accuracy [20].



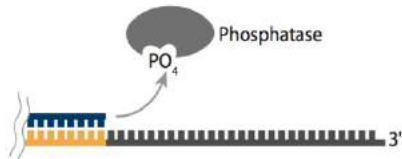
1. Prime and ligate



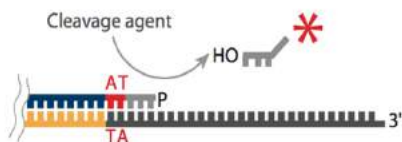
2. Image



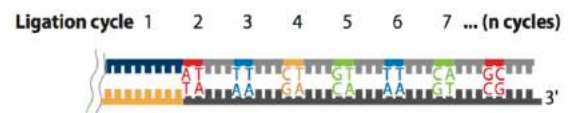
3. Cap unextended strands



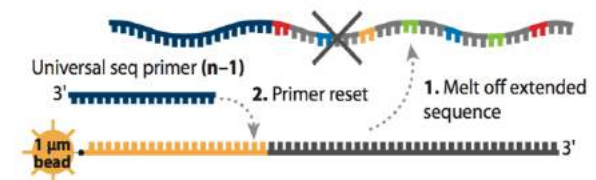
4. Cleave off fluor



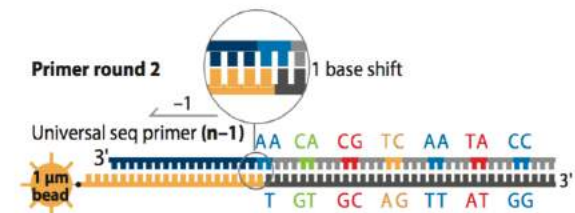
5. Repeat steps 1–4 to extend sequence



6. Primer reset



7. Repeat steps 1–5 with new primer



8. Repeat Reset with , n-2, n-3, n-4 primers






		Read position																																				
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	
Primer round	1	Universal seq primer (n) 3' 		●	●									●	●				●	●					●	●	●	●	●				●	●				
	2	Universal seq primer (n-1) 3' 		●	●								●	●				●	●					●	●				●	●			●	●				
	3	Universal seq primer (n-2) 3' 		Bridge probe		●	●					●	●				●	●				●	●				●	●				●	●				●	●
	4	Universal seq primer (n-3) 3' 		Bridge probe		●	●					●	●				●	●				●	●				●	●				●	●				●	●
	5	Universal seq primer (n-4) 3' 		Bridge probe		●	●					●	●				●	●				●	●				●	●				●	●				●	●
		● Indicates positions of interrogation Ligation cycle 1 2 3 4 5 6 7																																				

Figure 2.17: Outline of SOLiD sequencing technology adapted from [66].

2.3.4: NGS Raw Data File Formats and Quality Scores for Detected Nucleotides

The RNA-Seq experiment generates tens of millions of sequence tags, known as short reads, which can be encoded into different file formats depending on the NGS platforms such as the FASTQ [68] and FASTA/QUAL [69] formats. While encoding the short reads, not only the sequence of each read is preserved, but the quality value of detected nucleotides for each read is also determined. Although the quality scores across different platforms cannot be compared, all NGS platforms use a Phred-like score [70,71], which is logarithmically related to the probability that a base call is incorrectly identified (P).

$$Q_{PHRED} = -10 \times \log_{10} P \quad (2.2)$$

There are three types of quality scores that are used. The first scoring method is known as the Sanger quality score, which is used in Sanger FASTQ formats, and uses ASCII values from 33-126 to encode Phred scores from 0-93. Later by using logarithmic mapping, Solexa, Inc. (Illumina, Inc.), introduced another quality scoring method in which Solexa scores were used, ranging from -5 to 62 and represented by ASCII characters from 59 to 126 [68]. Solexa scores can be calculated by Equation 2.3.

$$Q_{Solexa} = -10 \times \log_{10} \frac{P}{1-P} \quad (2.3)$$

Equations 2.4 and 2.5 are used to convert Phred scores to Solexa-scale quality scores and vice versa.

$$Q_{PHRED} = 10 \times \log_{10} \left(10^{\frac{Q_{Solexa}}{10}} + 1 \right) \quad (2.4)$$

$$Q_{Solexa} = 10 \times \log_{10} \left(10^{\frac{Q_{PHRED}}{10}} - 1 \right) \quad (2.5)$$

More recently, Illumina, Inc. introduced a new format used from Genome Analyser Pipeline 1.3 onwards. In this format, the Phred scores ranging from 0-62 are represented by ASCII characters from 64-126 [68]. Table 2.2 shows a summary of the three different FASTQ formats.

Table 2.2: Summary of three quality score formats.

Description	ASCII characters		Quality scores	
	Range	Offset	Type	Range
Sanger standard	33-126	33	Phred	0 to 93
Solexa	59-126	64	Solexa	-5 to 62
Illumina	64-126	64	Phred	0 to 62

FASTQ format uses 4 lines to encode each read (see Figure 2.18). The first line begins with the character '@' and is followed by a sequence identifier and an optional description. The second line contains the order of nucleotides for the read. Line 3 begins with the '+' character and is optionally followed by the same sequence identifier. Lastly, line 4, which has the same length as line 2, indicates the quality of detected nucleotides for the read using ASCII characters [68,72].

```
@title      @Sequence Identifier
sequence    ACCCCAGGATCAACACTTCACATGCATTAGCAGAGAGAGATAAATCAA
+Optional text  +
Quality      =>=?A?<@B@A:?B?D;AC@@CAAAD<AAA:99?:@=?@B@77C><4
```

Figure 2.18: FASTQ format

FASTA format uses 2 lines to encode each read. The first line begins with the character '>' and is followed by a sequence identifier and an optional description. The second line contains the order of nucleotides for the read (see Figure 2.19).

```
@title      > Sequence Identifier
sequence    ACCCCAGGATCAACACTTCACATGCATTAGCAGAGAGAGATAAATCAA
```

Figure 2.19: FASTA format

Illumina encodes the reads and corresponding quality scores in the FASTQ format. Roche 454 encodes the reads in FNA format, which is a type of FASTA format, and also encodes the corresponding quality scores in a separate QUAL format which is also similar to FASTA format [73]. In contrast, since SOLiD output is based on colour space and not sequence space, this technology uses the CSFASTA (Colour Space FASTA) format to encode the sequence of a read, and QUAL format for the corresponding quality score [74].

2.4: Gene Expression Profiling Using NGS Technology (RNA-Seq)

The importance of mRNA in gene expression and its role in identifying the informative genes that cause disease was investigated in Section 2.1 and 2.2, where microarray was used to determine the expression of genes. Microarray is a reliable and robust method which has been proven over decades, and even considering the drop in cost of NGS technology, microarray is still more economical. However, microarray technology has several limitations. For instance, since microarrays are designed by hybridisation probes for which the sequence of the probes is already known, this means that this technology is ineffective at finding new genes, detecting structural variations, discovering transcripts, and analysing isoform composition. However, RNA-Seq technology can overcome these limitations [75]. Whilst comparative studies of microarray gene expression and RNA-Seq [21,76,77] suggest that the results of both platforms correlate well, a wider spectrum of gene expression levels could be obtained when RNA-Seq is used, resulting in a more detailed insight into gene expression. Further studies, prove that RNA-Seq outperforms microarray at discovering new isoforms [78,79], and at transcriptome profiling [80,81].

RNA-Seq or the Whole Transcriptome Shotgun Sequencing workflow consists of several steps (see Figure 2.20). First, the total RNA consisting of messenger RNA (mRNA), ribosomal RNA (rRNA), and other small RNAs is extracted from a cell. The next step is the isolation of RNA content. In this step, due to the fact that over 90% of total RNA is made of rRNA, which can hinder the detection of mRNA, it is necessary to remove rRNA. One solution for this issue is to use poly (A) to enrich mRNA, which can only be used if the interest is in analysing mRNA alone, as this method eliminates all other non-poly RNAs. Another solution is to use rRNA-depletion technique, such as using exonuclease to digest rRNA, or using subtractive hybridisation [82–84]. Furthermore, if a study is interested in other small RNAs, several strategies are available to enrich such small RNAs, either by using commercially available kits or performing size selection by using polyacrylamide gel electrophoresis [85].

The next step is fragmentation of RNA to reduce the chance of secondary structure formation, and to provide a homogeneous coverage of entire transcripts [75]. The final step is to convert single stranded RNAs to double stranded cDNAs by utilising a reverse transcriptase. This last step is done because most sequencing technologies are currently unable to sequence RNA itself. Once the cDNA libraries are formed, various NGS platforms can be used to sequence them as discussed in Section 2.3.

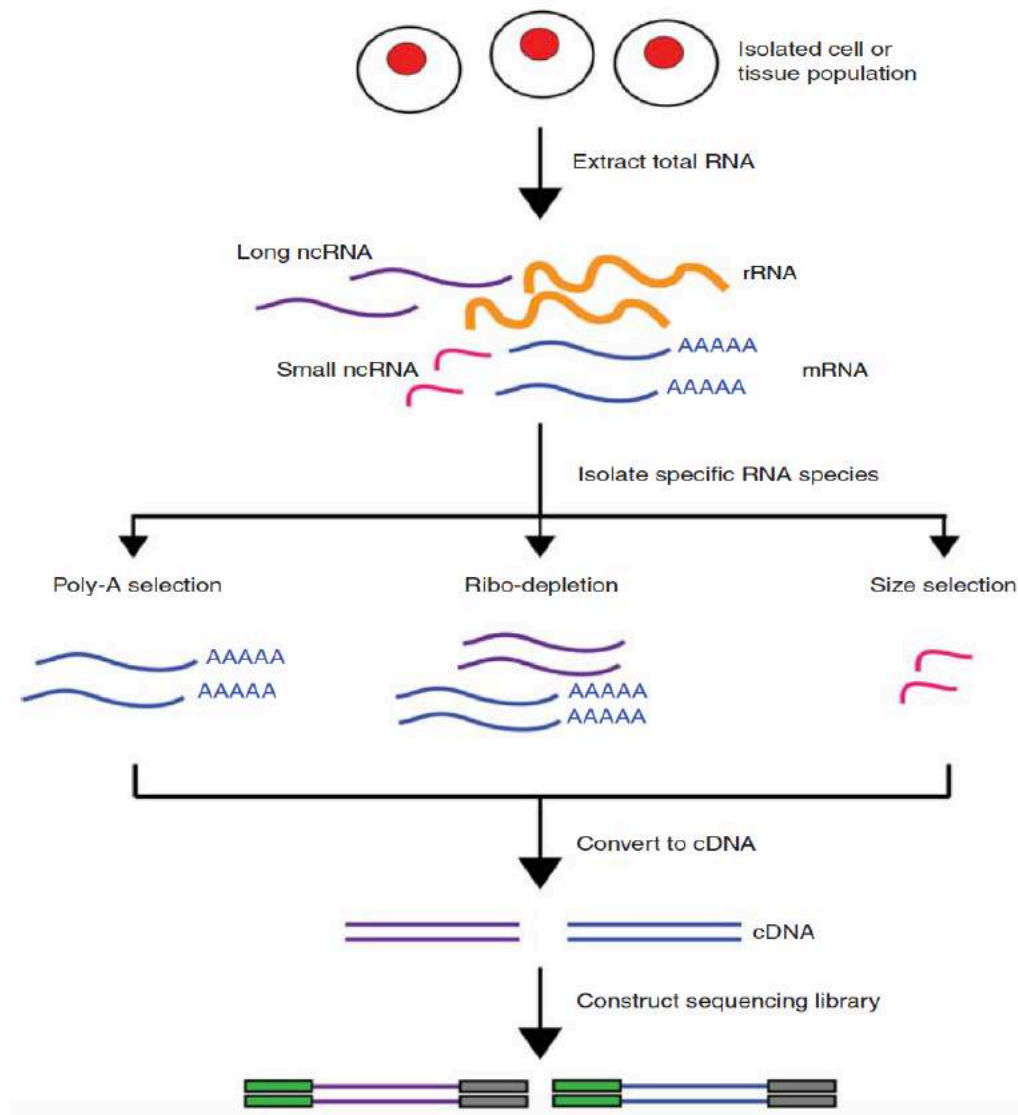


Figure 2.20: RNA-Seq procedure [86].

2.5: Analytical challenges for microarray and NGS data in respect to profiling and understanding diseases

Although gene expression profiling is a viable tool for diagnosis and prognosis of diseases, the analysis of microarray and NGS data is characterised as being very challenging [87]. In regard to microarray data, although only few samples are used, the expressions of thousands of genes are measured. This creates a challenge as the methods that could be implemented for analysis of microarray data needs to account for the nature of high dimensionality of such data [88]. To overcome this challenge and extract useful information among the pool of microarray data, machine learning techniques and various statistical approaches have been facilitated. These methods range from finding differentially expressed genes via statistical

methods to clustering and classification of diseases through machine learning techniques [46]. It is noted that over that last decade, the use of machine learning to classify diseases have become an area of intensive research [22]. However, high classification accuracy of diseases such as cancer is still highly challenging and more research is required where new methods could be implemented to further increase the classification accuracy.

With regard to NGS data, although the exploration of gene expression through RNA-Seq technique is more feasible, the analysis of such data is more computationally challenging when compared to microarray data [89]. For instance, a significant number of the short reads that are produced from a RNA-Seq experiment are map across splice junctions so that the task of mapping these reads to a reference genome is very challenging. Furthermore, after mapping these reads, the process of counting these reads over genomic locations inherits a significant challenge and one required to apply some statistical modelling such as discrete distributions to model these counts [90]. There exist several main challenges when analysing RNA-Seq as addressed in Chapter 6 later.

2.6: Summary

In this chapter a review of recent developments in measuring gene expression was presented. Initially, a brief overview of the biological aspects of the thesis such as gene expression phenomena was given. Afterwards, different types of microarray technologies such as cDNA and oligonucleotide microarray were explored. Then, different NGS technologies were investigated and a unique approach for each technique was explained in order to have a good understanding of how NGS data is produced. The common formats of NGS data such as Sanger, Solexa, and Illumina were then explained. Finally, the RNA-Seq technique utilising NGS technology to measure gene expression was introduced.

Chapter 3: Overview of Machine Learning Approaches for Microarray Data Analysis

3.1: Introduction

There are several steps for a successful microarray data analysis including design, pre-processing, inference, classification, and validation. Since each step plays an important role in the final results, there have been numerous studies to optimise each step. The design step is vital, as it determines the initial quality and quantity of the information to work with, and this step is carried out in a wet laboratory. Pre-processing is usually the first step for the analysis of microarray data, during which the images from microarray chips are processed and systematic variations are removed, followed by the transformation and normalisation steps. After pre-processing, depending on the purpose of the experiment, inference and/or classification of the data follows. Finally, the results from the previous steps are validated [46].

3.2: Design

Research suggests that the design of microarray experiments directly affects the efficiency and validity of the information obtained [91,92]. There are 2 factors that are essential to take into consideration when designing a microarray experiment.

The first factor is related to the importance of having biological replicates in the experiment. In general, there are two types of replicates, technical and biological replicates. Biological replicates refer to the replicates that are produced from different biological samples. In contrast, technical replicates are obtained from the same sample, but processed in a different

microarray experiment. One advantage of biological replicates compared to technical replicates is that the latter can only estimate the measurement variation between samples from different experiments. However, biological replicates not only can be used for this purpose, but they can also measure the variation between different biological samples. For instance, they can be used to find out which genes are differentially expressed between different samples [46].

The second factor is related to the number of required samples for a microarray experiment to provide enough information for a valid analysis. Several studies confirm that for a differential expression analysis using statistical inference, at least 5 biological replicates for each sample group are required [93,94]. With regards to a number of replicates for a classification purpose, a study by Dobbin and Simon [95] proposed a formula to calculate this number based on the relative sizes of different sources of variability.

3.3: Pre-Processing

As was discussed in Sections 2.2.1 and 2.2.2, as a result of microarray experiments, several images in TIFF format are produced that contain intensity signals. The first action towards a meaningful analysis is the pre-processing of these images and extraction of useful information to form a gene expression matrix [96,97]. Figure 3.1 shows the important steps for pre-processing microarray data.



Figure 3.1: Pre-processing of microarray data.

Image analysis is the first step in the pre-processing of microarray data, and deals with the process of extracting information from images. This provides the basis for further microarray analysis. This step involves quantifying spots on the microarray. To this end after identifying the spots on the microarray, first spot signal and background intensity are measured. Then based on these measurements, the initial intensity for each spot is calculated by subtracting the spot signal from the background intensity [98].

In the second step, the expression ratio for each gene is calculated. This is done by utilising the spot intensities of two samples and relating them by using a metric called expression ratio through the following equation.

$$T_k = \frac{R_k}{G_k} \quad (3.1)$$

where T_k is the expression ratio of gene k , R_k is the spot intensity of sample 1, and G_k is the spot intensity of sample 2. The expression ratio is a relevant way to represent expression differences. For example, genes that have equal levels of expression in two experimental conditions will have an expression ratio of 1. However, the interpretation of data from this method can be confusing when a gene has a higher or lower expression. For example, a gene that is highly expressed by a factor of 4, based on the formula $T_k = \frac{R_k}{G_k} = \frac{4}{1}$ has an expression ratio of 4. However, if it has a lower expression by a factor of 4, the expression ratio becomes 0.25 ($T_k = \frac{R_k}{G_k} = \frac{1}{4}$). Thus lower expression is mapped between 0 and 1 while higher expression is mapped between 1 and infinity. Logarithmic transformation is used to eliminate this inconsistency in the mapping intervals, where higher expression and lower expression are treated equally. For instance, if the expression ratio is 1, then $\log_2(1)$ equals 0 represents no change in expression. If the expression ratio is 4, then $\log_2(4)$ equals +2 and for an expression ratio of $\log_2(1/4)$ equals -2.

The next step is the normalisation of data. In the human genome, there are some genes known as housekeeping genes. The expression level of these genes should not change across different conditions. However, in some cases the data from the expression ratio suggests that an average expression ratio of such genes deviates from 1. This implies some sources of systematic variation that affect the measured expression levels of genes. In order to overcome this problem, the data needs to be normalised. There are several methods of normalisation, like total intensity normalisation, mean log centring, and linear regression. In these methods, a normalisation factor is calculated and then it is used to rescale the intensity of each gene [99].

After these pre-processing steps, the data can be represented in the form of a matrix, and each row in the matrix (see Figure 3.2) corresponds to a particular gene, while each column either corresponds to an experimental condition or a specific time point at which expression of the genes has been measured.

$$\mathbf{X} = \begin{array}{c} \begin{array}{ccccccc} \text{Sample 1} & \dots & \text{Sample } j & \dots & & & \\ \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,j} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & \dots & x_{2,j} & \dots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ x_{i,1} & x_{i,2} & \dots & x_{i,j} & \dots & x_{i,n} \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \dots & x_{m,j} & \dots & x_{m,n} \end{bmatrix} & \begin{array}{c} \text{Gene 1} \\ \vdots \\ \text{Gene } i \\ \vdots \end{array} \end{array} \end{array}$$

Figure 3.2: Gene expression matrix.

3.4: Unsupervised Classification

Data clustering, which also refers to unsupervised classification, is a way of finding similarities in data when no prior information on the structure of data is available [100]. In a clustering task, data is divided into groups in which data points within each group (cluster) are very similar to each other, yet different from other clusters. Microarray gene expression data can be clustered based on genes (row), samples (column), or both genes and samples which provides useful information for data visualisation and the interpretation of experimental results. Clustering based on both genes and samples referred to as bi-clustering [101] which is useful in uncovering functionally linked gene sets under different experimental conditions. Since genes that have similar expression pattern are grouped together in clustering methods, one can hypothesise that if two genes are within a similar cluster, the respective genes can be co-expressed and have a related function.

Despite the fact that there are several clustering methods available such as K-means, Fuzzy C-means and Hierarchical, all methods use a similarity measure to calculate the distance between data points, so that similarities and dissimilarities of all data points can be quantified and clustered respectively [102]. Among the several similarity measuring methods like Covariance, Manhattan Distance, Average Dot Product, Pearson Correlation Coefficient, and Euclidian Distance, the latter two are most commonly used.

In the Pearson Correlation, the linear association between gene a and b is calculated by Equation 3.2.

$$P_{ab} = \frac{\sum_{m=1}^M (x_{ma} - \bar{x}_a)(x_{mb} - \bar{x}_b)}{\sqrt{\sum_{m=1}^M (x_{ma} - \bar{x}_a)^2 \sum_{m=1}^M (x_{mb} - \bar{x}_b)^2}} \quad (3.2)$$

where x_{ma} and x_{mb} are respectively the gene expression for gene a and b in sample m and

$\overline{x_a}$ and $\overline{x_b}$ are the mean expression of genes a and b from all samples. The value of correlation between gene a and b , (Pab) can range from -1, which means a perfect negative correlation, to 1 which means a perfect positive correlation. If both genes appear to be independent of each other, the correlation value will be assigned to zero.

In Euclidian distance method, the distance between gene a and b is calculated by Equation 3.3.

$$Eab = \sqrt{\sum_{m=1}^M (x_{ma} - x_{mb})^2} \quad (3.3)$$

where x_{ma} and x_{mb} are respectively the gene expression for gene a and b in sample m . The Euclidian distance between gene a and b , (Eab) can range from 0 to ∞ .

3.4.1: K-means

K-means clustering is a simple and fast method that aims to partition n genes into k clusters, where data within a cluster is nearer to the centre of their cluster than other clusters. It is noted that the number of clusters (k) should be specified in advance. The means of clusters are updated, along with iterations. If that microarray data contains n genes with expression (x_1, x_2, \dots, x_n), and each x is a d -dimensional factor, n genes will be separated into k subsets with unknown centres ($\mu_1, \mu_2, \dots, \mu_k$), under the condition that $k \leq n$. The objective of the K-mean algorithm is to minimise the cost function H (see Equation 3.4), such that the centre of each cluster has the minimum aggregation distance between the centre of a cluster and the points within that cluster [103].

$$H = \sum_{j=1}^k \sum_{i=1}^n a_{ij} \|x_i - \mu_j\|^2 \quad (3.4)$$

where $\|x_i - \mu_j\|^2$ is the Euclidian distance between the i th gene (x_i), and the centroid for the j th cluster; a_{ij} is the membership value which is either one if x_i is assigned to j th cluster or zero otherwise. The K-mean algorithm clusters the data through the following iterative procedure:

Step 1: Random selection of cluster centres (μ_j) where the number of clusters are predefined (k).

Step 2: Assign each point (x_i) to its nearest cluster centre. This is done by determining the membership matrix a_{ij} using the following equation:

$$a_{ij} = \begin{cases} 1 & \text{if } \|x_i - \mu_j\|^2 \leq \|x_i - \mu_z\|^2 \text{ for all } j \neq z \\ 0 & \text{else} \end{cases} \quad (3.5)$$

Step 3: Compute the cost function H using Equation 3.4.

Step 4: Update the cluster centres as below:

$$\mu_j = \frac{\sum_{i=1}^n a_{i,j} x_i}{\sum_{i=1}^n a_{i,j}} \quad (3.6)$$

Step 5: Steps 2, 3 and 4 are cycled through continuously until they coincide with set values (e.g. the number of iterations).

3.4.2: Fuzzy C-means

Fuzzy C-means clustering relies on the basic idea behind K-mean clustering, with the difference that in the C-means method, each data point belongs to a cluster with a degree of membership grade. In K-means, each data point either belongs to a certain cluster or not. In other words, in fuzzy C-means, each data point can belong to more than one cluster with a degree of belonging specified by membership grades between 0 and 1. Fuzzy C-means also utilise a cost function and similar to K-means, its objective is to minimise the cost function [104]. The fuzzy C-means algorithm clusters the data by the following iterative procedure:

Step 1: Initialise the membership matrix $a_{i,j}^f$ with random values between 0 and 1, such that the constraint in Equation 3.7 is satisfied.

$$\sum_{j=1}^k a_{i,j}^f = 1 \quad \text{for } i = 1, 2, \dots, n \quad (3.7)$$

Step 2: The cluster centres are updated using the following expression:

$$\mu_j = \frac{\sum_{i=1}^n a_{i,j}^f x_i}{\sum_{i=1}^n a_{i,j}^f} \quad (3.8)$$

where $f > 1$ is the exponent of the membership values, and it is a real-valued number controlling the fuzziness of the clusters.

Step 3: Compute the cost function according to Equation 3.9:

$$H = \sum_{j=1}^k \sum_{i=1}^n a_{i,j}^f \|x_i - \mu_j\|^2 \quad (3.9)$$

Step 4: Compute a new membership matrix using following equation:

$$\mu_j = \frac{1}{\sum_{i=1}^k \left(\frac{\|x_i - \mu_j\|}{\|x_i - \mu_z\|} \right)^{\frac{2}{f-1}}} \quad (3.10)$$

Step 5: Steps 2, 3 and 4 are cycled through continuously until they coincide with set values.

3.4.3: Hierarchical Clustering

There are two types of hierarchical clustering approaches: agglomerative and divisive, both of which involve building some type of dendrogram or tree that reveals the relationships between the data objects. The agglomerative approach starts by assuming that each object belongs to its own cluster. Afterwards, it identifies which clusters are the closest to others using a distance metric. Each iteration of this approach creates bigger and bigger clusters at each level, until all data objects are put into one big cluster. In contrast, the divisive approach works exactly the opposite, where at the start all objects belong to one big cluster (see Figure 3.3). When the iteration starts, it finds the best division of the data objects, so that there is the highest similarity among objects within clusters, and the most dissimilarity between clusters. This process continues, until all objects are in their own clusters [5,105].

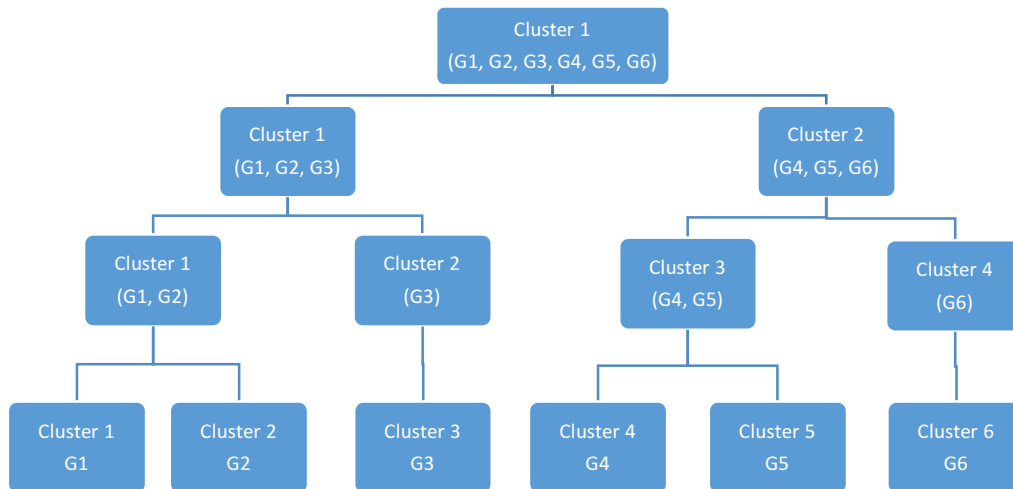


Figure 3.3: Chart of divisive hierarchical clustering scheme.

Hierarchical clustering partitions genes based on the measurement of distance. The most commonly used method for measuring distance in this context is Euclidean distance. The calculation of distance between a pair of sub-clusters depends on linkage criteria, which is a way of defining the similarity of clusters based on the similarities of cluster members. There are four linkage methods that can be used: single, average, complete, and the distance between centroids (see Figure 3.4), in which clusters are linked based on the similarity of the closest members, the average similarity, and the similarity of the furthest members [106].

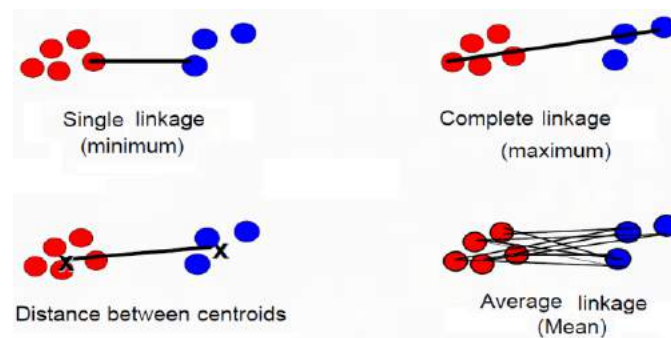


Figure 3.4: Linkage methods

3.4.4: Self-Organising Map

The Self-Organizing Map (SOM) which is based on neural network was developed by Kohonen [107]. SOM utilises a competition and cooperation means to attain unsupervised learning using winner takes all (WTA) algorithms. A SOM network consists of two layers including a Kohonen layer and an input layer (see Figure 3.5). In the Kohonen layer, neurons are organised in a geometric pattern, usually 2-dimensional lattice. Each neuron in the input layer is fully coupled with all neurons in the Kohonen layer and each has a weight vector, $w_i, i = 1, 2, \dots, h$ which is randomly initialised. h represents the number of neurons in the Kohonen layer. Also, each neuron in the Kohonen layer is linked to adjacent neurons by a neighbourhood relation. The objective of SOM is to discover a suitable mapping from the n dimensional input data ($x_i, i = 1, 2, \dots, n$) to a two-dimensional lattice configuration [108].

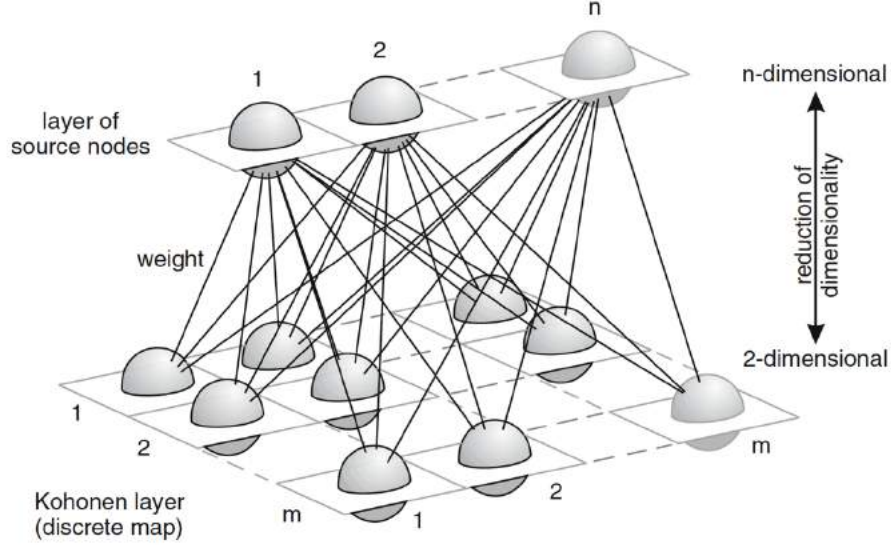


Figure 3.5: SOM neural network adapted from [108].

To this end, the Euclidean distances between input data and weight vectors are calculated. Then for each data vector the best match unit (BMU) is obtained through Equation 3.11, which refers to the unit that minimises the Euclidean distance between the input data and weight vectors [109].

$$BMU = \arg \min_j \|x_i - w_j\| \quad (3.11)$$

Once the BMU is chosen, this unit is then allowed to update its weight vector. Since in the Kohonen layer neurons are linked to adjacent neurons, when the BMU is chosen, all neighbouring neurons within a width and radius of BMU also will be updated. By defining a set of activated neuron adjacent to BMU as N_s , the activated neurons can update their weights at time t using Equation 3.12 [109].

$$w_j(t+1) = \begin{cases} w_i(t) + h(t) [x_i - w_i(t)], & i \in N_s \\ w_j(t), & i \notin N_s \end{cases} \quad (3.12)$$

where $h(t)$ is the neighbourhood function and can be calculated as below:

$$h(t) = \alpha(t) \exp\left(-\frac{r_{BMU} - r_i^2}{2\sigma^2(t)}\right) \quad (3.13)$$

where r_i denotes the location of neuron i on the grid map, $\alpha(t)$ is the learning rate, and $\sigma(t)$ is the kernel width function around BMU. Both $\alpha(t)$ and $\sigma(t)$ are monotonically shrinking over time. As the process continuous and new input vectors are given to the map, the neighbourhood radius and the learning rate progressively shrink to zero so that only BMU can be updated. The SOM algorithm clusters the data by the following iterative procedure [109]:

Step 1: The topology of SOM is defined and the weight of each neuron is randomly initialised ($w_i(0), i = 1, 2, \dots, h$).

Step 2: The distance between the input vector and the weights of each neuron is calculated and BMU is identified using Equation 3.11.

Step 3: The active radius around BMU is calculated which is then decreases over time.

Step 4: The weights of the BMU and the neurons within the active radius are updated using Equation 3.12.

Step 5: Steps 2, 3 and 4 are cycled through continuously until convergence.

3.4.5: Binarisation of Consensus Partition Matrices (Bi-CoPaM)

In order to improve reliability and robustness of clustering procedure, ensemble clustering methods such as graph-based and hypergraph-based methods [110], kernel-based methods [111], relabelling and voting [112], and non-negative matrix factorization [113] have been proposed. In these methods, the results of various clustering algorithms for the same dataset are merged to build a consensus clustering outcome.

More recently, the binarisation of consensus partition matrices (Bi-CoPaM) that is a tuneable consensus clustering method was proposed by Abu-Jamous *et. al* [114,115]. This clustering method takes into account various datasets while employs several single clustering algorithms to detect the subset of features which incessantly correlate among many clustering results [116]. This method outperforms conventional clustering algorithms in that each feature can be assigned to multiple clusters at the same time or not assigned to any clusters at all [117].

Assuming time series microarray datasets, Bi-CoPaM performs clustering independently for each dataset in the first stage and datasets are not combined. This means within each dataset

all features are homogenous as they have the same experimental design. However, in the next stages the created clusters from each dataset are merged in respect to memberships which is not influenced by the time profiles of the feature in their datasets to create one set of partitions. This method provides the infrastructure for multiple heterogeneous datasets to be analysed together [118]. Figure 3.6 illustrates the Bi-CoPaM flowchart.

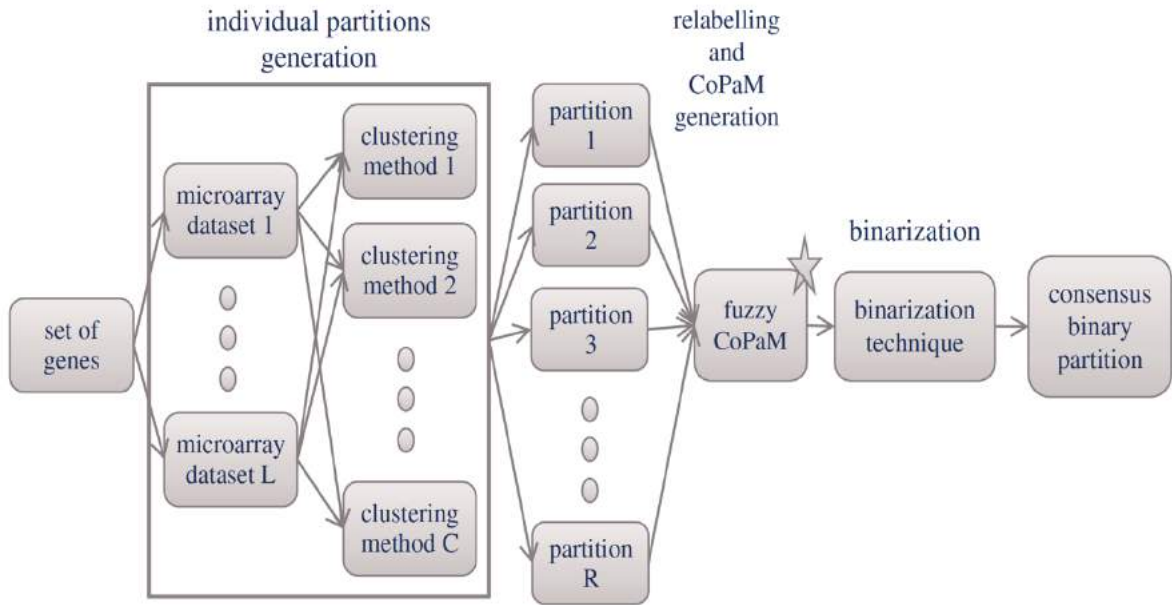


Figure 3.6: Flowchart of Bi-CoPaM adapted from [115]

Bi-CoPaM performs clustering through four main steps as follow:

Step 1: **Partitions generation.** In this step, R partition results, $\{U^1, U^2, \dots, U^R\}$, are created by facilitating R different clustering algorithms. Each U matrix consists of K rows corresponding to number of clusters and M columns corresponding to the number of genes. Each element in the matrix, U_{ij}^r , denotes the membership of the i th gene in the j th cluster based on the r th partition.

Step 2: **Relabelling.** The objective of relabelling is to make sure that the i th cluster in each generated partition correspond to each other by means of rearranging the clusters. The min-max approach is utilised to rearrange the clusters to achieve this objective as follow [115,117]:

- A) First a dissimilarity matrix, $D_{S \times S}$, is composed. In this matrix, each element (D_{ij}) denotes the dissimilarity between the i th row of the partition U and the j th row of the reference partition U_{ref} .
- B) Then in the D matrix the minimum of each column is calculated.
- C) Afterwards, the maximum value from the calculated minimums is selected by which the clusters from U and U_{ref} that have this maximum are coordinated to correspond to each other.
- D) Then the selected row that was used for matching the clusters in step C is removed from D matrix.
- E) Steps B to D are repeated till D matrix is empty and all clusters from U and U_{ref} are matched.

After relabelling the rearranged matrix of U is shown by \hat{U} which is presented as follow.

$$\hat{U} = \underset{\forall perm(U)}{argmax} \Gamma(U_{ref}, perm(U)) \quad (3.14)$$

where $perm(U)$ denotes the permutation of the rows of U and $\Gamma()$ is the similarity measure [114].

Step 3: CoPaM generation.

The R relabelled partition matrices are utilised to create a single fuzzy CoPaM in which each single element represents a fuzzy membership of a gene in a cluster based on the number of times that the genes appeared in that cluster. The membership value ranges from 0, representing absolutely no consistency to 1 which denotes absolute consistency. It is noted that the summation of fuzzy membership values for each given element across all clusters should be 1. To generate the CoPaM matrix the values of the first partition are used to instantiate an intermediate fuzzy CoPaM, U^{int} . The remaining partitions are then fused to the U^{int} one after another and in each step when a partition is fused, this partition is relabelled according to the U^{int} . Once all partitions are fused, the U^{int} is assigned as the CoPaM matrix U^* . If $U^{int(g)}$ denotes the intermediate matrix after g partition is fused, the generation of CoPaM can be carried out by the following three steps [115]:

- a) $U^{int(1)} = U^1$
- b) for $k = 2$ to R
 - 1. $\hat{U}^k = Relabeled(U^k, U^{int(k-1)})$
 - 2. $U^{int(k)} = \frac{1}{k} \hat{U}^k + \frac{k-1}{k} U^{int(k-1)}$
- c) $U^* = U^{int(R)}$

Step 4: **Binarisation.**

In general, the binarisation is performed in CoPaM so that each gene is included only in one cluster and excluded from other clusters. However, Bi-CoPaM generates a pseudo-partition matrix, B^* , with K rows corresponding to the number of clusters and M column which allows a gene to be included in multiple clusters, not assigned to any clusters or assigned to only one cluster by assigning a multiple 1s, no 1, or a unique 1 in columns accordingly. Several techniques have been proposed to generate the Bi-CoPaM including intersection binarisation (IB), maximum value binarisation (MVB), top binarisation (TB), different thresholding binarisation (DTB) [119]. In all methods, the binarisation status is monitored using two measurements namely M^{un} which denotes the number of genes that belongs to none of the clusters and M^{multi} that denotes the number of genes that belongs to multiple clusters. For instance, in IB approach binarisation leads to $M^{multi} = 0$ and $M^{un} \geq 0$ where a gene is assigned to a cluster if all partitions map this gene to all that cluster which can be mathematically expressed as follow:

$$b_{i,j}^* = \begin{cases} 1 & , u_{i,j}^* = 1 \\ 0 & , otherwise \end{cases} \quad (3.15)$$

In TB method, each gene is not only assigned to the maximum membership value cluster, but also assigned to other clusters where its membership values are within a definite variance δ less than the maximum which leads to $M^{multi} \geq 0$ and $M^{un} = 0$. TB method mathematically expressed as follow:

$$b_{i,j}^* = \begin{cases} 1 & , u_{i,j}^* - u_{k,j}^* \geq -\delta , K \geq k \geq 1 \ k \neq i \\ 0 & , otherwise \end{cases} \quad (3.16)$$

In DTP method, each gene is assigned to the maximum membership value cluster on a condition that the value of nearest candidate cluster is as far from the maximum as a minimum of a definite variance δ . This technique leads to $M^{multi} = 0$ and $M^{un} \geq 0$ when $\delta > 0$ and the value of M^{un} correlates with the value δ . DTP method mathematically expressed as in Equation 3.17.

$$b_{i,j}^* = \begin{cases} 1 & , \quad u_{i,j}^* - u_{k,j}^* \geq \delta, \quad K \geq k \geq 1 \quad k \neq i \\ 0 & , \quad \text{otherwise} \end{cases} \quad (3.17)$$

3.4.6: Unification of clustering results from multiple datasets using external specifications (UNCLES)

Abu-Jamous, et al., [118] proposed a method to examine multiple gene expression datasets at once by unifying the clustering results in order to discover a subset of genes that are co-expressed across all datasets while taking into account one of two types (type A and B) external specifications. Type A, is similar to that in Bi-CoPaM where the objective is to identify a subset of genes that are consistently co-expressed in all datasets. In contrast, type B allocate all datasets into two subsets of datasets, the negative subset, S^- , and the positive subset, S^+ . The objective of type B is to identify a subset of genes that are poorly co-expressed in S^- , while a consistent co-expression of the selected subset of genes can be observed in S^+ (See Figure 3.7).

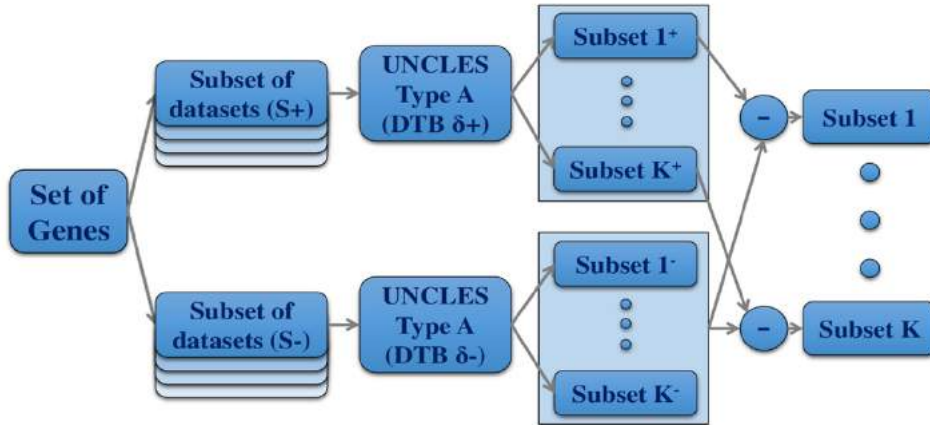


Figure 3.7: UNCLES flowchart with type B of external specifications adapted from [118]

Type B UNCLES is performed in several steps as follow. In the first step, Bi-CoPaM (type A) that utilises DTP binarisation with parameters δ^+ and δ^- is performed on both subsets of datasets S^+ and S^- respectively. Afterwards, the selected genes that resulted from Bi-CoPaM on the S^+ subset are excluded from the result of S^- . It can be noted that type B take advantages of two parameters (δ^+ and δ^-) compared to one parameter in type A. The parameter δ^+ regulates the tightness of clusters in S^+ subsets so that the co-expressed genes in this subset can be included in the final results and δ^- controls the how tight the clusters in S^- subset can be so that its co-expressed genes could be excluded from the final results. For instance, at a pair of (δ^+ and 0) will result in creating empty clusters as $\delta^- = 0$ means all the genes will be excluded from the final results [120].

3.5: Supervised Classification

In supervised classification, the objective is to design a class predictor to distinguish two or more classes of samples from each other (e.g. healthy vs cancerous). The class predictor is designed based on the currently available data from different diagnostic classes, which refers to training or learning samples. In this procedure, first the classifier is trained, and then the classifier is used to find the diagnostic class of new samples [7]. When designing a classifier, based on the available information within the training set of data, one needs to develop decision rules and mathematical formulas with a particular classifier design strategy, so that the classifier can make diagnostic or prognostic predictions. There exists several classifier design approaches that can be used for microarray gene expression classification, including Linear Discriminate Analysis (LDA) [7], k Nearest Neighbours (k-NN) [121], Support Vector Machines (SVM) [122], Multilayer perceptron (MLP) [123], and other types of Artificial Neural Networks [7].

LDA classification is based on the identification of linear combinations of features that are able to best distinguish between two classes of samples. This classification method is closely related to the analysis of variance method (ANOVA), and its objective is to maximise the ratio of between-class variance for datasets whereby a maximum separability between different diagnostic classes can be achieved [7]. k-NN classification is based on the concept of similarity measurement (e.g. Euclidian distance or Pearson's correlation) in which the distances between unknown samples (test samples) and known samples is calculated. Subsequently, the class membership of unknown samples is assigned based on k , the closest known samples. One of the advantages of k-NN is low computational consumption compared to other classification methods [124]. In this thesis, two of the most widely used classifiers, SVM and the MLP neural network are investigated in detail.

3.5.1: Support Vector Machine

SVM is an efficient classification method typically used for a two-class classification problem. SVM chooses a hyperplane, which provides the maximum separation distance in two classes [125]. Given some training data, a set of N points of the form (X_i, y_i) :

$$D = \{(X_i, y_i) | X_i \in \mathbb{R}^p, y_i \in \{-1, 1\}\}_{i=1}^N \quad (3.18)$$

where y_i is either 1 or -1 , indicating the class to which the point X_i belongs. Each X_i is a p -dimensional real vector. The objective is to find a maximum margin hyperplane that divides the points having $y_i = 1$ from those having $y_i = -1$. A hyperplane can be written by

expression $W^T X_i + b = 0$, where W is the normal vector to the hyperplane, X_i is the input vector ($X_i = X_1, X_2, \dots, X_N$), and b is the bias.

If the data is linearly separable, two hyperplanes can be selected, which provides the maximum separation distance for two classes (see Figure 3.8). The selection of the hyperplane is done in such a way that data is separated into two sections with a defined gap (margin) between separated data. The main objective is to maximise this gap to provide better classification results [126] .

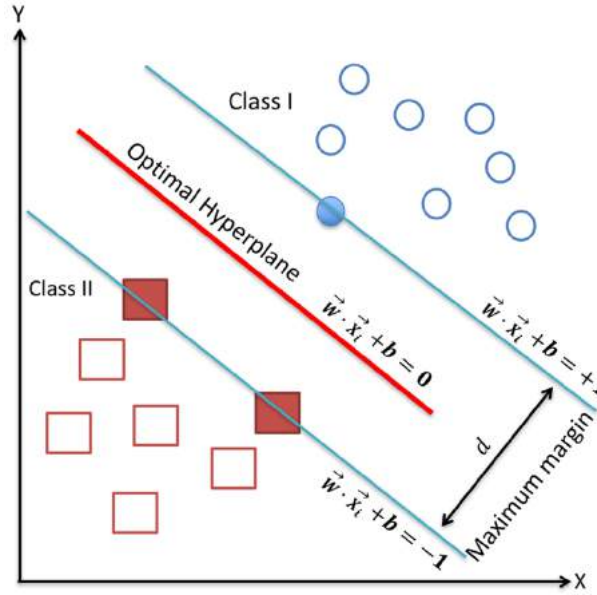


Figure 3.8: Support vector machine classifier.

In Figure 3.8, the blue lines are margin lines, and can be mathematically presented by Equations 3.19 and 3.20. The red line is the maximum-margin hyperplane that is mathematically formulated by Equation 3.21, whose position is in the middle of both margin hyperplanes. By using geometry, the distance between margin hyperplanes can be calculated by Equation 3.22.

$$W^T X_i + b = -1 \quad (3.19)$$

$$W^T X_i + b = +1 \quad (3.20)$$

$$W^T X_i + b = 0 \quad (3.21)$$

$$d = \frac{2}{\|W\|^2} \quad (3.22)$$

As it was discussed earlier, the objective is to maximise the distance between two margin hyperplanes which mathematically can be illustrated as Equation 3.23.

$$\max \frac{2}{\|W\|^2} \quad \text{or} \quad \max \frac{1}{\|W\|} \quad \text{or} \quad \min \|W\| \quad \text{or} \quad \min \frac{\|W\|^2}{2} \quad (3.23)$$

It is also crucial to prevent data points from falling into the margin. For this reason, the following constraint needs to be added:

$$\begin{cases} W^T X_i + b \geq 1, & y = 1 \\ W^T X_i + b \leq -1, & y = -1 \end{cases} \quad (3.24)$$

Equation 3.25 can be obtained by rewriting the above constraint:

$$y_i (W^T X_i + b) \geq 1, \quad 1 \leq i \leq n \quad (3.25)$$

In order to solve this optimisation problem that has such a constraint, the Lagrange method is utilised so that the constrained become unconstrained [127]. To this end, the problem can be stated in the Lagrange format as follows:

$$L(W, a, b) = \frac{\|W\|^2}{2} - \sum_{i=1}^N \alpha_i [y_i (W^T X_i + b) - 1] \quad (3.26)$$

Gradient with respect to W and derivation with respect to b will result in:

$$\nabla_W L(W, a, b) = 0 \Rightarrow W = \sum_{i=1}^N \alpha_i y_i X_i \quad (3.27)$$

$$\frac{\partial L(W, a, b)}{\partial b} = 0 \Rightarrow \sum_{i=1}^N \alpha_i y_i = 0 \quad (3.28)$$

Afterwards, these are substituted in the Lagrange formula as shown by the following equation.

$$L(a) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j X_i X_j \quad (3.29)$$

Under the constraining conditions:

$$\begin{cases} \alpha_i > 0 \\ \sum_{i=1}^N \alpha_i y_i = 0 \end{cases} \quad (3.30)$$

In Equation 3.29, by using a quadratic program, vector $\alpha_i = (\alpha_1, \alpha_2, \dots, \alpha_N)$ is created. It is noted that the majority of the α_i values are zero, and the value for α is only positive for support vectors. This means that one only needs to sum the equation over the support vectors. It is important to note that when classifying with SVM, at the same time the dimension of data is significantly reduced. As can be seen in Figure 3.8, the data has 15 dimensions while after using SVM the data effectively has 3 dimensions (pointed out by filled circle and squares shapes). Once the alphas that meet the criteria for support vectors are defined, they can be used and plugged into Equation 3.27 to calculate W . Then, the value for b can be calculated by the following expression:

$$y_n (W \cdot X_i + b) = 1 \quad (3.31)$$

Finally, the classifier can be designed as shown below:

$$f(X_{new}) = \sum_{i=1}^{sv} \alpha_i y_i (X_i \cdot X_{new}) + b \quad (3.32)$$

where sv is the number of support vectors, $X_i \cdot X_{new}$ is the dot products of the input vector sample and the unknown vector.

If the data is not linearly separable, then the nonlinear SVM is utilised by applying kernel trick [128]. In general, when kernel trick is applied on a pair of data, it can implicitly map this data to a higher dimensional space so that a linear classifier can be used to separate highly non-linear data. Training and classification process in nonlinear SVM is similar to that in linear SVM. The only difference between linear and nonlinear methods is that the nonlinear kernel function is used in nonlinear SVM. It is noted that when kernel trick is used the coordinates of the data in the higher dimensional space is not computed but rather the inner products of the data pairs is calculated which eliminates the computational power required for explicit computation of the coordinates [128,129]. In the case of non-linearly separable data the classifier can be designed as follows:

$$f(X_{new}) = \sum_{i=1}^{sv} \alpha_i y_i k(X_i, X_{new}) + b \quad (3.33)$$

where $k(X_i, X_{new})$ is a kernel function such as a polynomial, Gaussian or Hyperbolic tangent [130].

3.5.2: Multilayer Perceptron (MLP) Artificial Neural Network

- **Single layer perceptron:**

Artificial neural networks mimic biological neural networks like that of the human brain [131]. In biology, the fundamental unit of a biological neural network is a neuron, and in artificial neural networks the fundamental unit is an artificial neuron. One of the widely used models for an artificial neuron is McCulloch-Pitts (MP) model [132]. The model is constructed in such a way that it has one input layer of MP neurons feeding forward one output layer of neurons, referred to as a perceptron (see Figure 3.9). Each input has a weight (w). In a perceptron, an initially weighted sum of all its inputs is calculated and fed to a single variable function, which is also known as the activation function. The activation function then uses the information from the weighted sum to decide to fire or otherwise [133]. In other words, in its simplest form a perceptron is a network that can classify linearly separable patterns. To this end, initially the network should be trained in order to learn the values of the weights and biases to correctly respond to each input vector with the corresponding target classes.

Figure 3.9 shows a perceptron with m inputs ($x_1, x_2, x_3, \dots, x_m$), and corresponding synaptic weight for each input ($w_1, w_2, w_3, \dots, w_m$), a bias (b), activation function (f), and y is the output and can be mathematically presented by Equation 3.34.

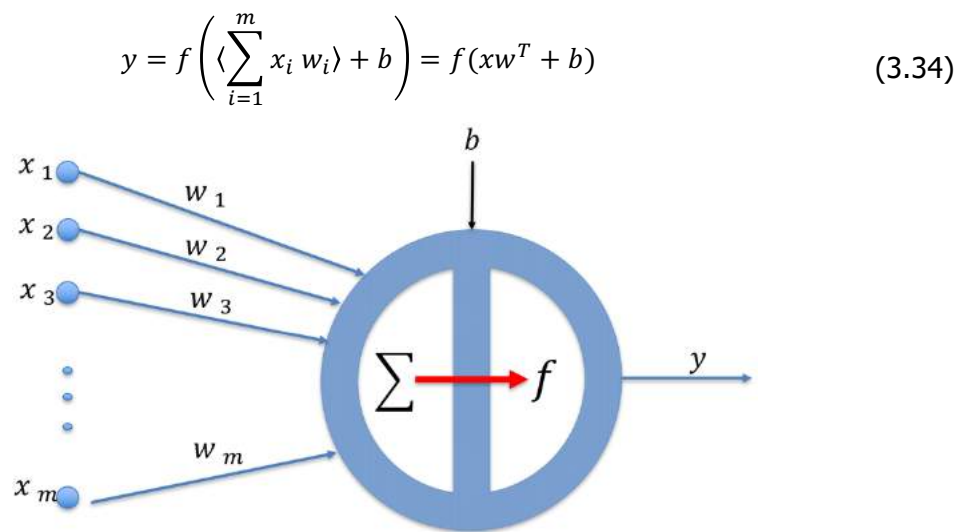
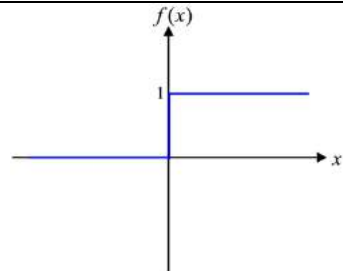
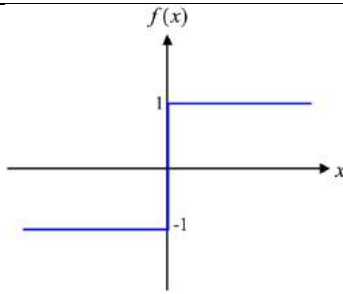
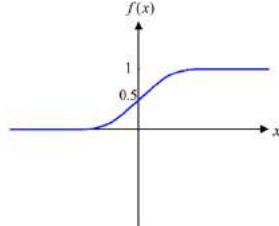
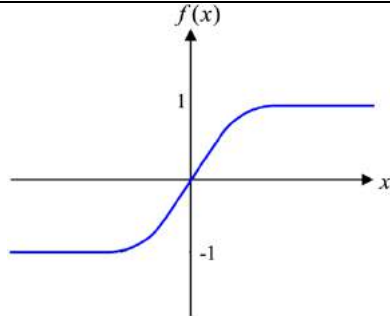


Figure 3.9: A perceptron with m inputs and a bias.

Table 3.1 depicts some commonly used activation functions, as well as their formulations, and the illustrations of the signal shapes.

Table 3.1 Activation functions

Function name	Formulation	Signal shape
Step	$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$	
Signum	$f(x) = \begin{cases} -1, & x < 0 \\ 1, & x \geq 0 \end{cases}$	
Sigmoid	$f(x) = \frac{1}{1 + e^{-\beta x}}$	
Hyperbolic tangent	$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	

For classification purposes, initially the network should be trained in order to learn the values of the weights and biases in order to minimise the error rate (error rate = desired output - actual output) [7]. A perceptron convergence algorithm can be used to train a single layer perceptron (SLP). In this algorithm the problem is solved in several steps that use the following parameters [134].

Input vector	$x(n) = [+1, x_1(n), x_2(n), x_3(n), \dots, x_m(n)]^T$
Weight vector	$w(n) = [b, (w_1(n), w_2(n), w_3(n), \dots, w_m(n))]^T$
Actual output	$y(n) = f(\sum_{i=0}^m w_i(n) x_i(n)) = f(w^T(n)x(n))$
Desired output	$d(n) = \begin{cases} +1, & x(n) \in \text{class1} \\ -1, & x(n) \in \text{class2} \end{cases}$

where n denotes the epoch number for applying the algorithm. It is noted that the input for bias (b) is equal to +1, and referred to as a synaptic weight of b in the weight vector. In the output, the summation operator starts at zero and $w_0(n)$ represents the weight of bias. The task of learning is done through four or five steps as follow:

1. Initialisation of weight vector in which $w(n) = 0$. Define the number of epochs to be performed ($n = 1, 2, 3 \dots, h$).
2. Activation of perceptron using input vector $x(n)$
3. For each instance in the input vector (with known class), the activation output of the signum function is computed using $y(n)$
4. Updating the weight vector using $\{w(n+1) = w(n) + \eta[d(n) - y(n)]x(n)\}$, where η is the learning rate parameter.
5. If the epoch number is less than h , increment epoch by one and go to step 2, otherwise stop.

- **Multilayer perceptron:**

The feedforward connection of at least two perceptrons leads to the formation of a multilayer perceptron (MLP) which can be used for classification of data even if the data is not linearly separable [135]. Each perceptron is fully connected to all perceptrons in the next layer, and a bias presents for each perceptron. In the MLP structure, the first and last layers are called input and output layers respectively, because they represent inputs and outputs of the overall network. The remaining layers are called hidden layers. Figure 3.10 illustrates a typical MLP configuration with two hidden layers. In this configuration, the input layer consists of N input features. The first hidden layer consists of 2 perceptrons, and each receives N inputs from the input features. The second hidden layer consists of 3 perceptrons, and each perceptron is fed by 2 inputs which are the outputs from the first hidden layer. Finally, the

output layer consists of one perceptron that has 3 inputs from the second hidden layer's outputs. All perceptrons have bias b .

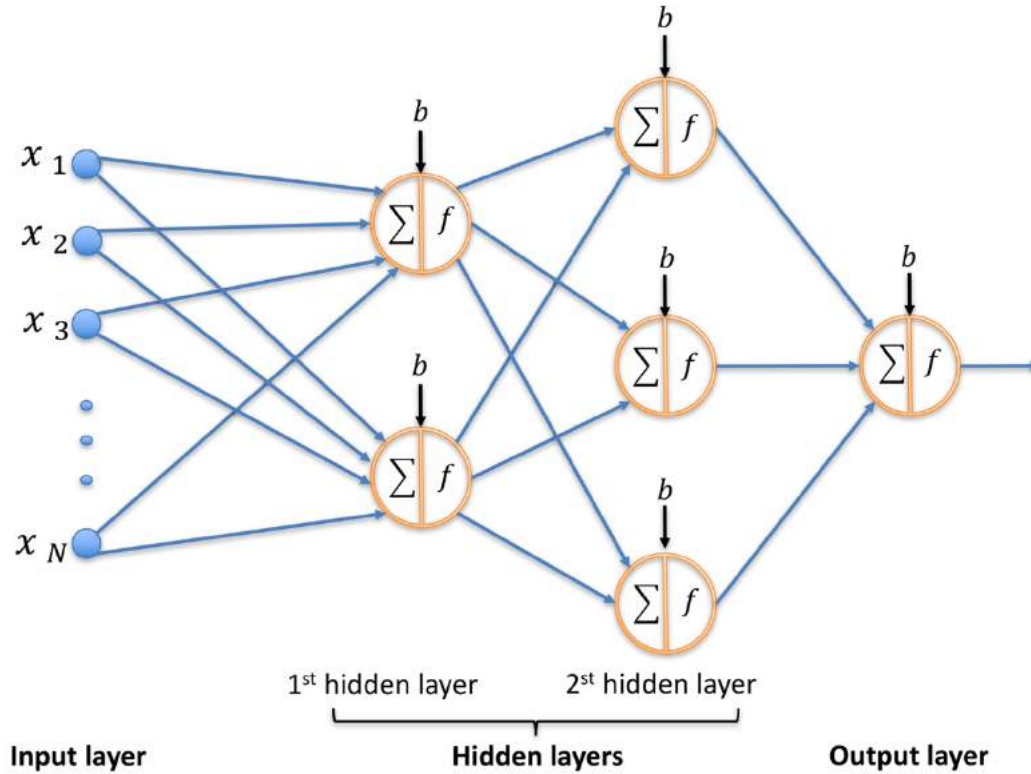


Figure 3.10: MLP Artificial neural network.

An activation function should meet several criteria, including being differentiable, monotonic, and continuous in order to be used in MLP learning [136]. This criterion is important because in later stages of training, one can apply gradient descent to find an optimum solution. Therefore, it is important to remember that both step and sign activation functions cannot be used. Between the hyperbolic tangent and sigmoid functions, the latter one is most widely used [137].

The training (learning) is usually done by error back propagation algorithms, which are based on error correcting learning rules [138]. Compared to SLP, where all inputs are directly connected to the neuron that produces the output, in MLP the inputs have indirect effects on the output. The main idea in MLP is to calculate the error rate at the output layer (layer L) and then back propagate them to the perceptron in the previous layer ($L-1$), after which the weight is updated accordingly to minimise the errors. The back propagation algorithm is performed through several steps. Initialisation of weight vectors is the first step, and the number of epochs to be performed ($n = 1, 2, 3 \dots, h$) is also defined at this step. The second step is the forward computation step, where the output activation functions for each layer and the error for the output layer is calculated. In this respect, we denote l for layer ($1 \leq l \leq L$),

i for inputs ($0 \leq i \leq d^{l-1}$), and j for outputs ($0 \leq j \leq d^l$). For the first step the objective is to determine the parameters of $w_{ij}^{(l)}$ as the synaptic weight of neuron i in layer l [134]. The function that is used to calculate the output signal for each layer ($x_j^{(l)}$) is shown by Equation 3.35.

$$y_j^{(l)}(n) = \theta(s_j^{(l)}) = \theta \left(\sum_i^{d^{l-1}} w_{ij}^{(l)}(n) y_i^{(l-1)}(n) \right) \quad (3.35)$$

where θ is the activation function (sigmoid), n is the epoch number and $y_i^{(l-1)}$ is the output activation function of neuron i in the previous layer $l - 1$. To solve the problem, the stochastic gradient decent (SGD) method can be applied. The error can be defined as a function of weight vector $e(w_{ij}^{(l)})$. Therefore, to apply SGD we need the gradient of $e(w_{ij}^{(l)})$ as follows:

$$\nabla e(w_{ij}^{(l)}) = \nabla e(W) = \frac{\partial e(W)}{\partial w_{ij}^{(l)}} \quad (3.36)$$

In order to acquire the gradient of the error, we can rewrite $\nabla e(W)$ as follows:

$$\frac{\partial e(W)}{\partial w_{ij}^{(l)}} = \frac{\partial e(W)}{\partial s_j^{(l)}} \times \frac{\partial s_j^{(l)}}{\partial w_{ij}^{(l)}} \quad (3.37)$$

where $\frac{\partial s_j^{(l)}}{\partial w_{ij}^{(l)}} = y_i^{(l-1)}$, the value of which is already calculated by Equation 3.35. Therefore,

one only needs to calculate $\frac{\partial e(W)}{\partial s_j^{(l)}} = \delta_j^{(l)}$. To calculate the δ for the final layer where $l = L$

and $j = 1$ the following expression is used:

$$\delta_1^{(L)} = \frac{\partial e(W)}{\partial s_1^{(L)}} \quad (3.38)$$

For the final layer, it is noted that using the mean squared error, $e(W) = (x_1^{(L)} - y_k)^2$, where $y_1^{(L)} = \theta(s_1^{(L)})$ and y_k is a constant and presents the desired value. Therefore, δ for the output layer can be computed by the following expression:

$$\delta_1^{(L)} = \theta'(s_1^{(L)}) \quad (3.39)$$

where θ' is the derivative of the sigmoid activation function [134].

The third step, backward computation, aims to calculate the error of previous layers through back propagation using the error from the output layer using the following equations:

$$\delta_1^{(l-1)} = \frac{\partial e(W)}{\partial s_i^{(l-1)}} = \sum_{j=1}^{d^{(l)}} \frac{\partial e(W)}{\partial s_j^{(l)}} \times \frac{\partial s_j^{(l)}}{\partial x_i^{(l-1)}} \times \frac{\partial x_i^{(l-1)}}{\partial s_i^{(l-1)}} \quad (3.40)$$

$$\delta_1^{(l-1)} = \sum_{j=1}^{d^{(l)}} \delta_j^{(l)} \times w_{ij}^{(l)} \times \theta'(s_i^{(l-1)}) \quad (3.41)$$

Finally, the weights are updated using equation 3.42:

$$w_{ij}^{(l)} = w_{ij}^{(l)} - \eta x_i^{(l-1)} \delta_j^{(l)} \quad (3.42)$$

This step terminates an epoch, so that if the number of epochs is less than h , steps 2 and 3 are repeated, otherwise the final value of $w_{ij}^{(l)}$ is returned as the final weights.

Although there are several training algorithms based on back-propagation, such as gradient descent [139], conjugate gradient [140], Bayesian regularisation [141], resilient [142], scaled conjugate gradient [143], and Levenberg-Marquardt [144], the last one is the most widely used.

3.6: Feature Selection

During microarray analysis, the number of genes is significantly higher than the number of samples [12,13] and classification to a high level of accuracy is challenging, due to large number of genes and small sample size [14,15]. This concept refers to as the curse of dimensionality which is a term that was introduced by Belham to explain the challenge initiated by the exponential expansion in volume related to adding extra dimension to Euclidian space [145]. In order to overcome this problem, gene selection mechanisms have been introduced, by which only the most important genes are selected and used for classification purposes [16–19]. There are several advantages to this process of minimising the number of genes, and only selecting the meaningful genes which are more predictive during classification. By having

fewer genes, not only is the processing time for classification significantly decreased, but the chance of misclassification is also reduced. Furthermore, inputting a high number of genes into the classifier can cause the classifier to be over-fitted [146].

Gene selection methods, based on their interaction with the classifier, can be categorised into three approaches: filter methods, wrapper methods, and embedded methods [146,147]. Filter methods assess the relevance of genes by only looking at the general characteristics of the data, and ignoring the impact of selected genes on the classification performance [148]. Wrapper gene selection initiates a search procedure in the space of possible gene subsets. The selected genes are then evaluated based on their power to improve classification accuracy [149–151]. In the embedded gene selection method, feature selection is linked to the classification stage, but this connection is much stronger than in the wrapper method. This is because gene selection in embedded methods is included in the classifier construction, and the classifier is used to provide a criterion for feature selection [152,153] (see Figure 3.11). More recently, evolutionary algorithms have been utilised for gene selection within the framework of wrapper methods [154,155].

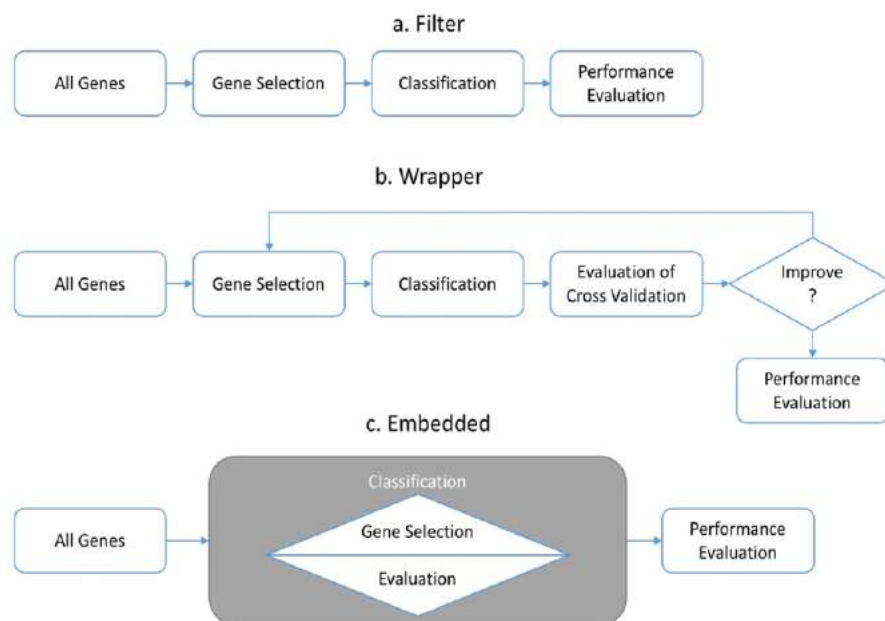


Figure 3.11: Feature selection methods.

Each gene selection approach has advantages and disadvantages [146]. For instance, although the filter method is simple and computationally efficient, its performance lags behind other approaches. This is because the classifier performs independently, and is not involved in the selection of genes [156]. Conversely, while the wrapper and embedded methods, which incorporate the gene selection task into the classification task, can achieve higher classification

accuracy, they suffer from scalability problems due to their high computational cost and are not practical for large datasets [157,158].

3.7: Overfitting

As discussed in Section 3.5, in a classification task there exist two main phases namely training and testing. In the training phase, the classifier model is build using training data and in the second phase the model is evaluated using test data. It is important to note that the test data should not be used in the training phase, otherwise the result of validation would be optimistic. The main aim when building a classifier model is not only to perform well on the training data, but to be able to generalise this model to perform well on the test data and other unseen data [159]. Overfitting is a phenomenon that occurs when a model is too complex (too many parameters) that the model memorises the training data rather than learn to generalise from the data. In other word, overfitting happens when fitting the data in the model more than it is warranted [160].

As illustrated in Figure 3.12, initially as the number of parameters in a model increases the error rate of classifier decreases for both training and test data. However, after the 5th parameter is included in the model, the error rate for test data starts to increase while the training data exhibits a low classification error. Therefore, if overfitting takes place the model performs very well on the training data, however this model would have a poor prediction power when applied to the test data due to the lack of generalisation [161]. The impact of overfitting could be in a higher magnitude on the classification performance for unseen data if the training data consists of stochastic noise.

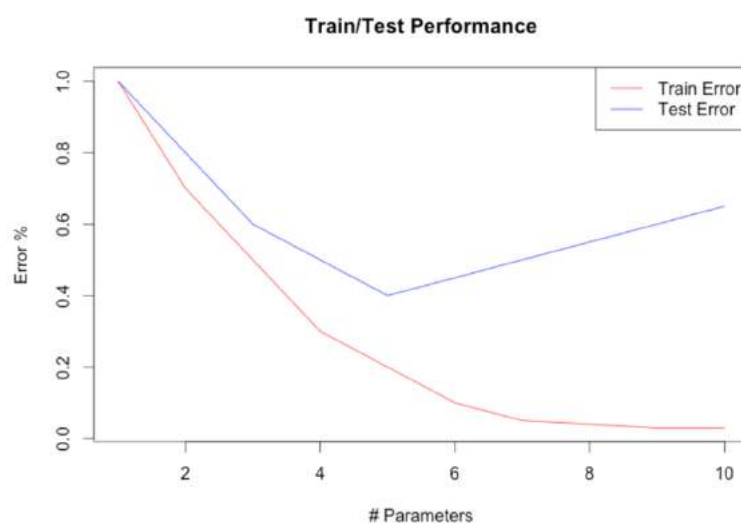


Figure 3.12: Train and test performance when changing the number of parameters in the classifier model adapted from [162].

Overfitting can be prevented if methods such as hold out validation, k-fold cross validation, or leave one out cross validation (LOOCV) are implemented in the model. These methods basically determine the point where further training will not result in enhancing the generalisation power of the classifier. In general, in a cross validation task the data is split into two parts where the training is done on one part and validation is performed on the other part. Therefore, the principle of cross validation emphasis on separating a part of data from the training stage to validate the performance of the model on this part of data which is not seen by the model before. Cross validation is widely accepted in machine learning society where it is being use for model selection [163].

In holdout validation, the data is split into two parts (e.g. 70% - 30%). The training is usually performed on the higher chunk of data (70% of data), then the model is evaluated on the remaining part (30 % of data). Since splitting the data is a random process, one usually tend to repeat the splitting several times and consequently repeating training and evaluation several times. Then report the final accuracy as the average accuracies that obtained from all repetitions [164].

K-fold cross validation is commonly used technique for assessing the prediction performance of a classifier model. In this method data is split into k equal chunks where the training is performed on k-1 chunks and the testing is carried out on the remaining chunk. This process is repeated k times, where each time a new chunk is chosen for test phase and the remaining k-1 chunks for training. Therefore, testing is performed on all chunks separately. The final accuracy of the model is determined by averaging the accuracies in each iteration. Similar to hold out validation, to acquire a robust estimate of the classification performance the k-fold cross validation should be ran multiple times while reshuffle the data each time. Then the final estimate is reported as the average accuracies obtained in each iteration. LOOCV is a special case of k-fold cross validation where k is equal to the number of samples [164].

The K-fold cross validation can be used for model selection. In the above, the cross validation using k-fold cross validation was discussed. In a model selection task using k-fold the task is somewhat similar to holdout method whilst here the splitting of data refers to a "three-way holdout". In this approach, the data initially is split into two parts namely test and training set. The test set is preserved for the final evaluation of the model. The training set is then used for k-fold cross validation. Once the training and validation is done, the performance is accessed based on the test data [163].

3.8: Summary

In this chapter, the steps required for microarray analysis were described including: pre-processing, clustering, and classification (see Figure 3.13). It was explained that the design step is a crucial step towards a successful analysis. Then the importance of pre-processing of microarray data was explored, and it was concluded that, first, systematic variation of microarray images should be removed; and the importance of transformation and normalisation of the data before starting the main analysis was pointed out. Furthermore, unsupervised and supervised classification methods were described. It was elaborated that unsupervised analysis can provide useful information for data visualisation and the interpretation of experimental results; several clustering methods such as K-means, C-means, hierarchical, SOM, Bi-CoPam and UNCLES clustering methods were investigated. Then the importance of supervised classification methods in class prediction was mentioned, and methods such as SVM and the MLP artificial neural network were briefly explained. Afterwards, the vital role of feature selection before classification was investigated. Finally, the pitfalls of overfitting and how to account for it were discussed

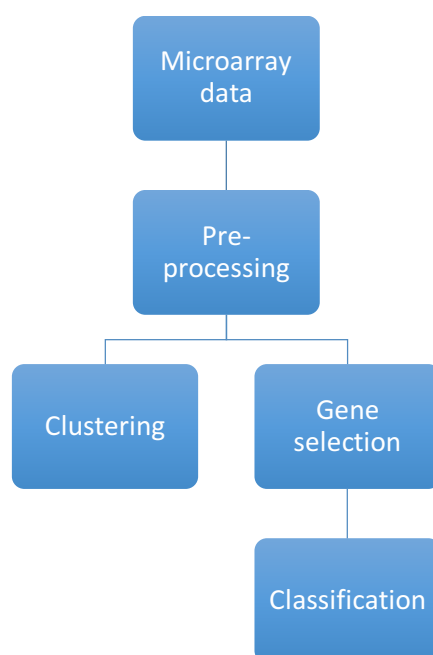


Figure 3.13: Microarray data analysis.

Chapter 4: Effects of Data Clustering Prior to Gene Selection on Cancer Classification

4.1: Introduction

In order to enhance classification performance, two main areas including gene selection and classifier design are important to be investigated. Furthermore, it is referenced from the literature that grouping data (clustering) has also been implemented for microarray data analysis in a number of investigations [165]. The main characteristic of such approaches is that there is no prior information on the group structure of the data, and frequently used in microarray analysis to facilitate the visual display of experimental results.

In this chapter, the effects of gene clustering prior to gene selection on classification accuracy is investigated. In this context, the aim of clustering applications is to partition n genes (total number of genes) with m dimension (m sample) into a given number of clusters. Once the data is clustered, a set of genes is selected based on gene ranking across all clusters for classification purposes. In order to fully investigate the effects of clustering on classification accuracy, not only are conventional clustering methods such as K-means, fuzzy C-Means and hierarchical methods used, but some optimisation algorithms are also utilised including PSO, GA, and COA for clustering purposes. Furthermore, a novel optimisation algorithm called COA-GA is proposed for clustering tasks.

4.2: Optimisation Based Clustering Techniques

In order to investigate the effects of optimisation based clustering methods on classification performance, three optimisation algorithms, specifically GA [25], PSO [166], and COA [24] were used. A new hybrid optimisation algorithm, COA-GA, was also developed, merging the recently invented COA and the traditional GA algorithms for data clustering. In the following subsections, first the design of the cost function for clustering tasks will be given. Then, details of optimisation algorithms including GA, PSO, COA will be described. Finally, the newly proposed hybrid COA-GA algorithm are explained.

4.2.1: Proposed Cost Function

In an optimisation problem, the optimisation algorithm iterates until a fitness function (cost function) conforms to a threshold set beforehand. Therefore, in order to use the optimisation algorithm for the purpose of clustering microarray gene expression data, a cost function needs to be defined with the objective to minimise the distance between data within each cluster, while maximising the distance between clusters. The design of the cost function is depicted below.

1. An evolutionary algorithm randomly creates an initial population from microarray data (POP).

$$POP = \begin{bmatrix} X_{1,1} & \cdots & X_{1,p} \\ \vdots & \ddots & \vdots \\ X_{m,1} & \cdots & X_{m,p} \end{bmatrix}$$

where m is the population size which is supplied by evolutionary algorithm, and p is the product of number of samples (s) in the data set and number of clusters (c).

2. For each row of the POP matrix steps 3 to 9 are repeated.
3. Candidate centres are acquired by reshaping a row of the POP matrix with dimension of $c \times s$ as follows:

$$CandidateCenters = \begin{bmatrix} X_{1,1} & \cdots & X_{1,s} \\ \vdots & \ddots & \vdots \\ X_{c,1} & \cdots & X_{c,s} \end{bmatrix}$$

4. Then the distances between each increment of data (gene) and candidate cluster centres are calculated:

$$Distance = \begin{bmatrix} X_{1,1} & \cdots & X_{1,c} \\ \vdots & \ddots & \vdots \\ X_{i,1} & \cdots & X_{i,c} \end{bmatrix}$$

where i is the number of genes.

5. "Minimum values" for each gene in the *Distance* matrix is found, thereby a gene will be assigned to the cluster which has the minimum value for that gene.
6. Distance between all clusters is calculated and assigned to variable B .
7. Distance between all clusters is calculated as follow. First by using "*dist*" function of MATLAB the Euclidian distance between all clusters are calculated. This produce a distance matrix whose dimension is $c \times c$. Then the upper triangular part of this matrix is selected and the sum of columns is calculated which results in a vector that has c elements. Finally, the sum of this vector is computed which results in a single value. This value is assigned to variable B .
8. In order to ensure each suggested cluster centre contains at least one gene, a term called "*penalty*" is defined. If a cluster contains at least one gene, this term will become zero, otherwise it will be $10e4$ (essentially to skip unsuitable cluster centres).
9. Finally, the cost for the selected row is calculated as below:

$$Cost = \text{sum}(\text{minimum values}) + \frac{1}{B} + \text{penalty} \quad (4.1)$$

10. Since steps 3-9 are repeated m times, a matrix containing cost values whose dimension is $m \times 1$ will be acquired and return.

The objective is to supply this cost function to an evolutionary algorithm whereby the chosen population by the algorithm can be examined in terms of their profit value. The cost function will be updated by each iteration of algorithm. Through some initial experiments, it was observed that the cost function could be minimised up to 100 iterations beyond which no further cost minimisation was observed. Therefore, in order to control the computational time, it was decided to run all optimisation algorithms 100 times in this research and compare their performances.

4.2.2: Genetic Algorithm (GA)

The genetic algorithm is an evolutionary computing method that was first introduced by John Holland in 1975 [25]. Since then, this algorithm has been used for solving many computational problems that require searching through a huge number of possibilities for solutions. By using a genetic algorithm, many different possibilities are explored simultaneously in an efficient way. The foundation for the method comes from the behaviour of living organisms in nature. In biology, an enormous set of possibilities lies in a set of possible genetic sequences, and the desired solutions are highly fit organisms that can survive and

reproduce in their environments. In a genetic algorithm, a potential solution to the problem is named as a chromosome. In the first step of this algorithm, an initial set of chromosomes, referred to as initial population is selected. From this population, some individuals are randomly opted to transfer to the next generation without any change occurring to them through a natural selection process [167].

In the selection method, a fitness function (cost function) is used for evaluating the quality of every chromosome. According to the principles of evolution, chromosomes with higher fitness scores tend to remain for producing offspring [168]. Therefore, the probability that an individual is transferred to the next generation is defined by how good its fitness function is. Each gene in the chromosome represents a specific characteristic. If all the chromosomes are transferred to the next generation, the next generation's properties will be identical to the previous generation's properties. However, in reality, this is not the case. In fact, two events take place in chromosomes. The first event is mutation, where the random substitution of some nucleotides within each chromosome occurs. The role of mutation is to increase the possibility of exploring untouched areas of the design space, preventing premature convergence. The number of genes that undergo mutation is very low (less than 10%). However, this random variation is really important. The second event is crossover, where the beginning of one chromosome sticks to the end of another chromosome (genetic recombination). The number of genes that undergo crossover is higher than that for mutation [169]. The cost minimisation plot is acquired to visualise how GA minimises the cost function over 100 iterations. The pseudo-code of GA is given below.

1. Initialise population.
2. Calculate fitness.
3. Sort fitness value of the population.
4. Choose the best fit solution to be the parental pair for reproduction.
5. Crossover the chromosomes at a random position using single point crossover.
6. Mutation.
7. Evaluate cost for the new offspring's chromosomes and mutated chromosomes.
8. If the number of iterations is less than 100, go to step 2.
9. Save the best profit so far as the 'best answer'.

4.2.3: Particle Swarm Optimisation (PSO)

Particle Swarm Optimisation was first introduced by Eberhart in 1995, and was intended for simulating the social behaviour of the movement of organisms in a bird flock or fish school.

This kind of action is an automatic and interactively updated system [166]. PSO has already been implemented in many research areas, such as function optimisation, artificial neural networks, and fuzzy system control.

Particle swarm optimisation (PSO) is a method that optimises an issue by iteration, which tries to achieve the best result for a given function. In PSO algorithms, a population (or swarm) consists of several particles or candidate solutions. These particles are moved around in the search space based on its own memory and information received from other particles in order to find the best solution [170]. Like genetic algorithm, a fitness function is used for determining the fitness value of each particle. The fitness value also needs to be optimised. In the progress of movement, the position of each particle is adjusted by the change of velocity, which is based on its own experience and particles around it. The velocity represents the rate at which a particle changes its position. This kind of movement can be represented as:

$$v_i(k+1) = v_i(k) + \gamma_{1i}(p_i - x_i(k)) + \gamma_{2i}(G - x_i(k)) \quad (4.2)$$

$$x_i(k+1) = x_i(k) + v_i(k+1) \quad (4.3)$$

where, v and x are the velocity and position of i th particle; k represents iteration level; p is the best position found by i th particle (personal best); G accounts for the best position found by the swarm (global best); γ_{1i} and γ_{2i} are random numbers on the interval $[0,1]$ applied to i th particle. The above movement iteration will stop after a set number of times [171]. The cost minimisation plot is acquired to visualise how PSO minimises the cost function over 100 iterations. The pseudo-code of PSO is given below.

1. Initialise population (n particles).
2. Calculate the fitness of each particle.
3. Position of the best-fit particle is chosen as the global best position.
4. Move all of the particles towards the global best position.
5. For each particle, if (fitness of current position < fitness of personal best) then personal best = current position.
6. Update personal best position for each particle.
7. Global best fitness value is retained.
8. If number of iteration is less than 100, go to step 2.
9. Save the global best from 100 iterations as the 'best answer'.

4.2.4: Cuckoo Optimisation Algorithm (COA)

COA is a population-based optimisation algorithm that was proposed by Rajabion in 2011 [24] that was inspired by the life of the cuckoo bird. The cuckoo's behaviour in laying eggs is unique in the sense that a cuckoo never builds its own nest when laying eggs, and instead uses other birds' nests to lay its eggs. In doing so, if the cuckoo's eggs are similar to the host's eggs, it is likely that the cuckoo's eggs will hatch and become mature cuckoos. If the cuckoo's eggs are discovered by the host bird, the foreign eggs will be destroyed. In the COA algorithm, each egg in a nest represents a potential solution and each cuckoo represents a successful new solution. The objective of the COA is to find the nest with the highest probability of an egg's survival. Therefore, the more eggs that survive after being placed in a host nest, the greater the level of profit assigned to that nest. When the time comes for the migration of the newly matured cuckoos, they will move towards the best nest with the highest survival rate, and lay eggs within a radius of it. This radius is known as the egg laying radius (ELR), and can be calculated by Equation 4.4.

$$ELR = \alpha \times \frac{\text{Number of current cuckoo's eggs}}{\text{total number of eggs}} \times (var_{high} - var_{low}) \quad (4.4)$$

where α is an integer, intended to control the maximum value of ELR, var_{low} and var_{high} are respectively the minimum and maximum values in the gene expression dataset.

Around 10 % of the laid eggs are sufficiently dissimilar to the nest's eggs and are killed by the host bird; the rest would remain until they turn into mature cuckoos and form societies. Each society has its own habitat area to live in [172]. When the time for egg laying approaches for newly matured cuckoos, they migrate towards the best habitat among all societies (goal point). As illustrated in Figure 4.1, when cuckoos move towards the goal point they can deviate by ϕ , where ϕ is a number between $\frac{\pi}{6}$ and $-\frac{\pi}{6}$, in which case they can only fly λ amount of the distance between the current habitat and the goal point (d) in that iteration, where λ is a random number between 0 and 1 [24].

Cuckoo's living area

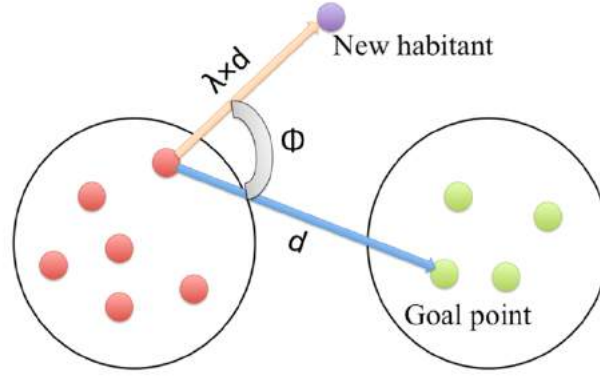


Figure 4.1: Immigration of a cuckoo towards goal habitat.

When all cuckoos have migrated toward the goal point and new habitats have been specified, each cuckoo is allocated some eggs. Then after the number of eggs dedicated to each bird is considered, an egg laying radius (ELR) is calculated for each cuckoo, and this step concludes one iteration in the algorithm. In the new iteration, the new egg laying process starts. Due to the fact that there is always equilibrium in any birds' population, a number N_{Max} is provided in the COA algorithm to control and limit the maximum number of live cuckoos in the environment [173]. After some iterations, all the cuckoo populations move to the optimum habitat. This habitat will produce the maximum profit, and there will be the least egg losses in this best habitat [173]. The functional immigration formula in COA is defined as:

$$X_{Nexthabitat} = X_{Currenthabitat} + F \times (X_{goalpoint} - X_{Currenthabitat}) \quad (4.5)$$

where F is a parameter that causes deviation. The cost minimisation plot is acquired to visualise how COA minimises the cost function over 100 iterations [174]. The COA algorithm follows the steps listed below [24].

1. Initialise cuckoo habitats with some random points in the profit function.
2. Dedicate some eggs to each cuckoo.
3. Define ELR for each cuckoo.
4. Allow cuckoos to lay eggs inside their corresponding ELR.
5. Kill the eggs that are recognised by host birds (if two eggs are in the same position).
6. Let eggs hatch and chicks grow.
7. Evaluate the position of each newly grown cuckoo (profit value).
8. Limit cuckoos' maximum number in the environment, and kill those who live in the worst habitats.

9. Assign the current maximum profit using the cuckoo with highest profit value.
10. Cluster cuckoos (using k-mean), find the best group, and select goal habitat.
11. Let new cuckoo population immigrate toward goal habit.
12. Get the position of all cuckoos and their profit values and update maximum profit.
13. If the number of iteration is less than 100, go to step 2.
14. Save the positions of cuckoo with highest profit value as the 'best answer'.

In the COA algorithm like other optimisation algorithms there are few parameters that are important to set as follow. Default values are used.

- Number of initial population.
- Maximum number of cuckoos to control how many cuckoos can live at the same time in each iteration.
- Minimum number of eggs for each cuckoo.
- Maximum number of eggs for each cuckoo.
- λ variable to controls distance between the current habitat and the goal point.
- Radius coefficient to control the egg laying radius.
- Number of k-means clusters.

4.2.5: Proposed COA-GA Algorithm for Clustering

A new algorithm is developed by hybridising COA and GA. Figure 4.2 shows the flowchart of the COA-GA algorithm. First, the COA chooses the best population (pop 1) as discussed in Section 4.2.4, and the profit value (fitness value) is calculated for this population. 50% of the chosen population undergoes the crossover operation, which is intended to prevent premature convergence as it creates more solutions in a given population [169]. After crossover, a 20% mutation is applied to the population, which increases the chance of discovering a better solution by maintaining diversity within the population. Crossover and mutation are important aspects of GA, increasing the possibility of exploring untouched areas of the solution space in each iteration of the algorithm, which COA alone could not reach. The output of these processes is termed population 2 (pop 2), and the profit value is determined for this population.

Pop 2 and pop 1 profit values are compared, and the population with the higher profit value is retained (population with better positioning) and input into the next iteration of the algorithm. This process is repeated 100 times, refining the cost function with each iteration.

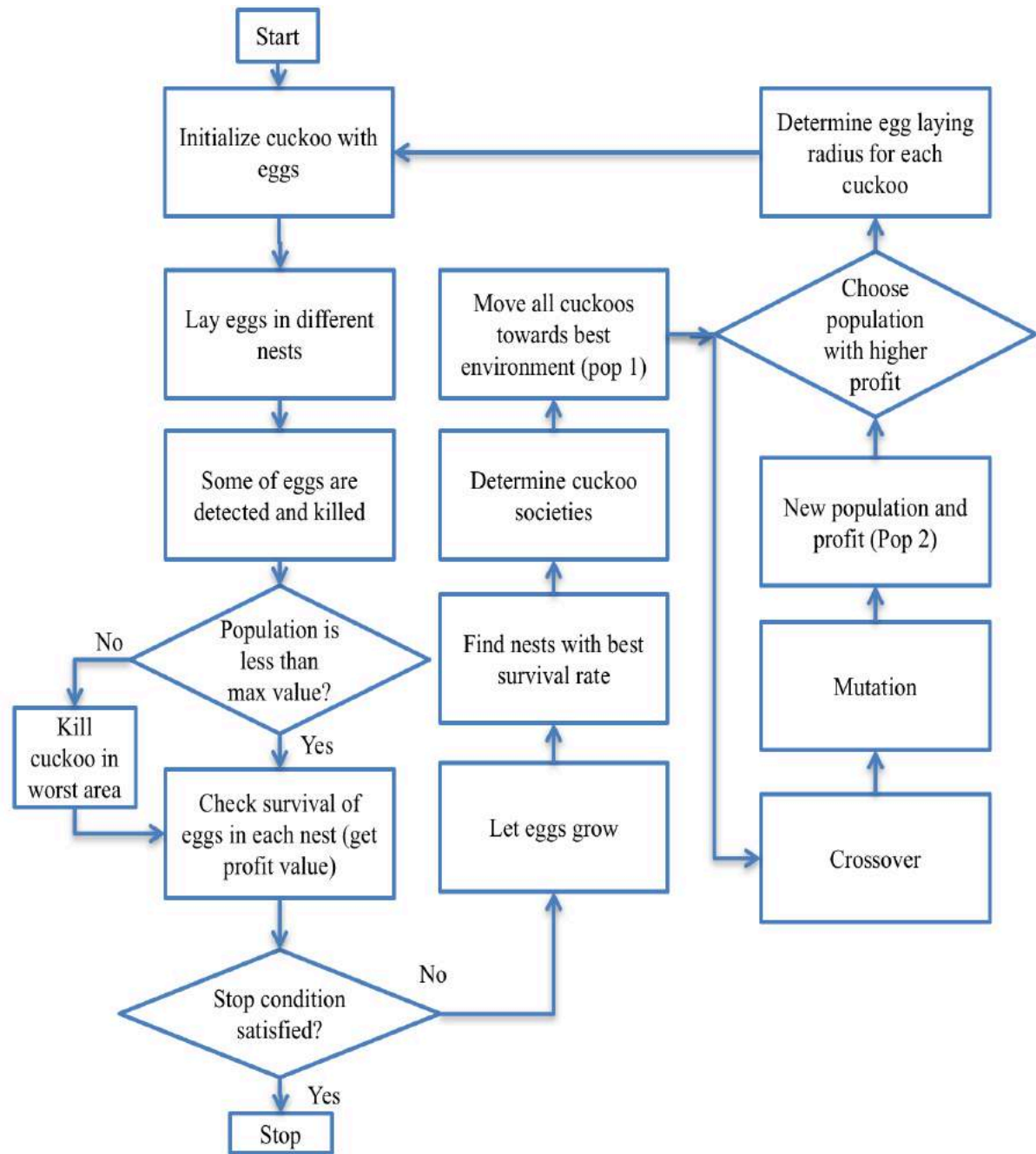


Figure 4.2: Flowchart of COA-GA.

4.3: Gene Ranking and Selection

As it was discussed in Section 3.6, gene selection is an essential task in microarray data analysis, due to the fact that only small numbers of genes are informative for each cancer type, and the presence of other genes reduces the classification accuracy. In this chapter, in order to facilitate a quick search and therefore reduce the computational time, the filter method of gene selection is used to score the genes. In this method, gene scoring is performed

by utilising a signal-to-noise ratio (SNR) criterion. The general expression for SNR is shown in Equation 4.6.

$$SNR = \frac{Signal}{Noise} = \frac{Distance}{Variance} \quad (4.6)$$

There are several SNR-based ranking methods such as the signed Fisher discriminant ratio (Signed.FDR), Fisher discriminant ratio (FDR), symmetric divergence (SD), and T-statistics that can be used for gene ranking [118, 188, 189]. A summary of these methods is provided below.

Name	Criterion	
Signed-FDR	$\frac{\mu_i^+ - \mu_i^-}{\sigma_i^+ + \sigma_i^-}$	(4.7)
FDR	$\frac{(\mu_i^+ - \mu_i^-)^2}{(\sigma_i^+)^2 + (\sigma_i^-)^2}$	(4.8)
SD	$\left(\frac{(\sigma_i^+)^2}{(\sigma_i^-)^2} + \frac{(\sigma_i^-)^2}{(\sigma_i^+)^2} \right) - 1 + \frac{1}{2} \left(\frac{(\mu_i^+ - \mu_i^-)^2}{(\sigma_i^+)^2 + (\sigma_i^-)^2} \right)$	(4.9)
T-test	$\frac{\mu_i^+ - \mu_i^-}{\sqrt{\frac{(r_i^+)^2}{N^+} + \frac{(r_i^-)^2}{N^-}}}$	(4.10)

where μ_i^+ and σ_i^+ are the mean and standard deviation respectively of the class (I) of gene i , and μ_i^- and σ_i^- are the mean and standard deviation of the class (II) of gene i respectively. N^+ and N^- are the number of samples in class (I) and class (II) respectively. $(r_i^+)^2$ and $(r_i^-)^2$ can be calculated based on following equations:

$$(r_i^+)^2 = \frac{\sum_{k \in class I} (x_{ik} - \mu_i^+)^2}{N^+ - 1} \quad (4.11)$$

$$(r_i^-)^2 = \frac{\sum_{k \in class II} (x_{ik} - \mu_i^-)^2}{N^- - 1} \quad (4.12)$$

In this chapter, after clustering is performed the genes in each cluster were ranked using symmetric divergence method (See Equation 4.9) which is a filter based ranking technique. The total number of best genes to be selected from all clusters is set to be $N_g = 25$. The number of best genes to be selected from each cluster is calculated using Equation 4.13.

$$N_g^k = \text{round} \left((N_g - q) \frac{\sum_{i=1}^{m_k} F_{score}(X_i, t)}{\sum_{i=1}^m F_{score}(X_i, t)} \right) + 1 \quad (4.13)$$

where N_g^k is the number of best genes selected from cluster k , N_g is the total number of best genes to be obtained from all clusters, q is the number of clusters, m is the total number of genes, m_k is the number of genes in cluster k and $F_{score}(X_i, t)$ is the criterion used for gene ranking (Equation 4.9). In this study, at least one gene is selected from each cluster. In this respect, the number of clusters is subtracted from the total number of required genes and then the number of genes in each cluster is added by one [165].

4.4: Classification and Performance Evaluation

In most cases, before classification the data is divided into two partitions: test and training sets. For both the training and test data, hold out validation is applied to get accurate classification result. After partitioning the data, the classifier trains itself by using the training data, and then tests its prediction power across the test data. Finally, the prediction outcome is compared to the testing target, and as a result the accuracy of the classifier is calculated.

4.4.1: Classification Methods

In this study, the SVM (see Section 3.5.1) and MLP (see Section 3.5.2) artificial neural networks are used as the classifiers. In the case of the SVM classifier, the build of the hyperplane is based on the structural risk minimisation principle. The error rate of the learning machine for the test data is bounded by the training error rate, as well as one term that depends on the Vapnik-Chervonenkis (VC) dimension [177,178]. The input data is first mapped in the feature space, in relevance to the kernel function. Then the system automatically searches for an optimised linear division [179].

In the case of MLP, the classifier has 25 inputs that are fed by the 25 selected genes; one hidden layer consisting of 30 neurons, and one output. Sigmoid and pure linear activation functions are used for the hidden and output layers respectively as the activation functions. The Levenberg–Marquardt algorithm [180] is used for training purposes, and the maximum

number of iterations is set to be 100. 70% of the data is used for training and 30% is used for testing the classifier performance. In order to reduce the effects of random selection on the training and testing data, the neural network has been trained and tested 100 times, where in each iteration, different training and testing data sets were used.

4.4.2: Performance Evaluation

After the classification task, the performance of both classifiers is evaluated. The evaluation is carried out in the forms of sensitivity, accuracy, and specificity. There are 4 possible outcomes from the classifier. The first possibility is a true positive (TP), which refers to the case that a diseased sample is correctly diagnosed. The second possibility is a false positive (FP), in which a healthy sample is incorrectly identified as a diseased case. The third possibility is a true negative (TN), which indicates the case where a healthy sample is correctly spotted. The final possibility is a false negative (FN), which refers to the case that the diseased sample is incorrectly identified as healthy [165]. The percentage value for the evaluation criteria (sensitivity, specificity and accuracy) can be calculated using equations 4.14-4.16.

$$Sensitivity = \frac{n_{TP}}{n_{TP} + n_{FP}} \times 100 \quad (4.14)$$

$$Specificity = \frac{n_{TN}}{n_{TN} + n_{FN}} \times 100 \quad (4.15)$$

$$Accuracy = \frac{n_{TP} + n_{TN}}{n_{TP} + n_{TN} + n_{FP} + n_{FN}} \times 100 \quad (4.16)$$

where n_{TP} , n_{TN} , n_{FN} and n_{FP} correspond to the number of TP , TN , FN and FP respectively as a result of the classifier test stage.

4.5: Investigating the Effects of Conventional Clustering Methods on Classification Performance

4.5.1: Methods

The general methodology used in this section is illustrated in Figure 4.3. The gene expression data for prostate and leukaemia cancer was used for this investigation. First, the data was indexed by using the available information on the classes of data (e.g. healthy vs cancerous). The data was indexed in two groups and stored separately in two matrixes

referred to as *IndexClass1* and *IndexClass2*. After the genes were indexed, 3 conventional clustering methods (K-means, fuzzy C-means, and hierarchical) were utilised to partition genes based on their similarity. The number of clusters was pre-defined.

For each dataset, the following steps were performed six independent times, each time choosing a different number of clusters ($k = 1, 2, 3, 4, 5, 6$). If data is clustered into one, this means no clustering was performed.

1. Data is clustered into k cluster.
2. Symmetric divergence (see Equation 4.9) was used for gene ranking.
3. The top 25 ranked genes were selected using Equation 4.13.
4. The selected genes were then fed to the SVM (see Section 3.5.1) and MLP (see Section 3.5.2) classifiers.
5. Classification performances for both classifiers were evaluated in terms of sensitivity, accuracy, and specificity as explained in Section 4.4.2.

Note that the performance of clustering is assessed based on their effect on the classification performance. Changing the number of clusters results in selection of different genes due to the method of gene selection (Equation 4.13) which is affected by the gene ranking within each cluster and the number of genes in each cluster. Therefore, each clustering method will result in different selected genes which subsequently will result in different classification performances.

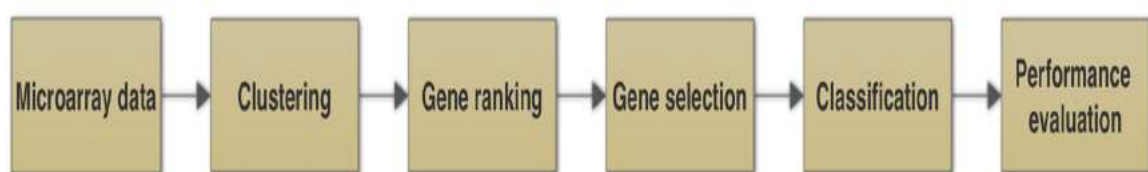


Figure 4.3: Proposed microarray data analysis procedure.

4.5.2: Results

Basic information on the datasets used in this research is listed in Table 4.1, including the number of genes, samples, and the two classes.

Table 4.1: Basic information of microarray data.

Dataset	Number of genes	Samples	Class1	Class2
Leukaemia	7,129	72	48 (ALL)	25 (AML)
Prostate	12,600	102	50 (Normal)	52 (Cancerous)

At the first stage of microarray analysis, data was clustered in order to find any hidden connections throughout it without any annotations. In order to investigate how each clustering algorithm distributes genes into different clusters, the number of clusters was set to two. After running each clustering algorithm, the number of genes in each cluster was observed (see Table 4.2). It is noteworthy that genes are not equally partitioned, and each clustering method partitions genes differently. The differences between the numbers of genes across two clusters are more pronounced when the K-means and fuzzy C-means algorithms are used. In contrast, the information suggests that hierarchical clustering divides genes into clusters more equally.

Table 4.2: Number of genes in each cluster when data is clustered into two groups

	K-Means		C-Means		Hierarchical	
	Cluster 1	Cluster 2	Cluster 1	Cluster 2	Cluster 1	Cluster 2
<i>Number of genes for Prostate</i>	117	12483	184	12416	6189	6411
<i>Number of genes for Leukaemia</i>	140	6989	135	6994	1603	5526

In order to investigate the effects of clustering on classifier performance, different conventional clustering methods (K-means, fuzzy C-means, and hierarchical) were used. For each type of clustering, the classifier performance has been tested by partitioning data into different amounts of clusters (1, 2, 3, 4, 5, and 6 clusters), where 1 cluster means no clustering was used. After clustering, the top 25 genes were selected using the filter method of gene selection (Equation 4.13) and fed to the classifiers.

To investigate the performance of the MLP classifier for the selected genes, the selected genes (25 genes) were fed to the MLP and mean sensitivity, specificity, and accuracy were calculated. Also, standard deviations for these terms were calculated. In the case of the leukaemia dataset, as can be seen from

Table 4.3, when no clustering (1 cluster) was used, sensitivity, specificity, and accuracy are found to be 81.1%, 83%, and 89.8% respectively. This yielded more accurate results compared to when clustering was used, apart from the case of K-means clustering when data was partitioned into 6 clusters, and 82.1%, 83.5%, and 90.1% were acquired for sensitivity, specificity and accuracy respectively. In respect to the prostate cancer dataset (See Table 4.4), in some clustering cases such as using K-means with 3 clusters, C-means with 2 clusters, and hierarchical with 5 clusters, a slightly better classification performance for the MLP classifier was observed. This was compared to the case when no clustering was used, in which 84.9%, 89.1%, and 87% were acquired for sensitivity, specificity, and accuracy respectively. The results from both datasets suggest that clustering may not necessarily enhance the MLP classifier performance.

Table 4.3: MLP classifier performances including the mean sensitivity, specificity, accuracy, and standard deviation (SD) for leukaemia.

		Number of clusters					
		1	2	3	4	5	6
		Mean/SD	Mean/SD	Mean/SD	Mean/SD	Mean/SD	Mean/SD
K-means	Sensitivity	81.1/15.2	80.3/16.4	80.9/16.2	79.1/17.3	81/15.0	82.1/13.3
	Specificity	83/12.9	82.7/13.5	82.6/13.7	82.8/14.0	82.5/13.6	83.5/12.2
	Accuracy	89.8/8.2	87.6/8.9	89.1/8.5	88.9/9.3	89.6/8.3	90.1/7.9
C-means	Sensitivity	81.1/15.2	81/15.4	80.4/16.4	80.9/16.0	81.1/15.2	79.9/17.9
	Specificity	83/12.9	82.9/12.9	82.1/13.3	82.5/13.1	82.9/13.0	82.3/13.7
	Accuracy	89.8/8.2	89.3/8.4	89.1/8.9	88.4/9.4	88.9/9.1	88.4/9.4
Hierarchical	Sensitivity	81.1/15.2	81/15.4	80.9/16.4	80.2/16.9	79.9/17.0	80.5/16.6
	Specificity	83/12.9	82.3/13.4	82.9/13.0	83/12.9	82.6/13.2	81.9/14.2
	Accuracy	89.8/8.2	88.2/9.4	89.4/8.0	89/8.6	88.8/9.1	89.5/7.9

Table 4.4: MLP classifier performances including the mean sensitivity, specificity, accuracy, and standard deviation (SD) for prostate cancer.

		Number of clusters					
		1	2	3	4	5	6
		Mean/SD	Mean/SD	Mean/SD	Mean/SD	Mean/SD	Mean/SD
K-means	Sensitivity	84.9/12.1	84.7/12.5	85.1/11.6	84.7/12.5	84.2/12.8	83.6/13.1
	Specificity	89.1/9.1	88.6/10.4	89.8/8.7	88.5/10.5	88.9/10.1	88.3/10.9
	Accuracy	87/10.9	86.7/11.3	87.1/10.7	86.9/11.0	84.6/12.3	86.8/11.5
C-means	Sensitivity	84.9/12.1	85.2/11.5	84.2/12.6	84.3/12.5	83.6/13.0	83.9/13.4
	Specificity	89.1/9.1	89.7/8.9	87/11.3	88.6/10.1	89.1/9.2	87/11.3
	Accuracy	87/10.4	87.1/10.3	85.2/11.5	86.5/11.0	85.1/11.6	85.9/11.1
Hierarchical	Sensitivity	84.9/12.1	84.3/12.6	84.2/12.9	84.8/12.2	85/11.5	83/13.7
	Specificity	89.1/9.1	88.1/10.6	89/9.4	87.5/11.1	90.3/8.1	89/9.4
	Accuracy	87/10.4	86.9/10.5	86/11.0	86.5/10.7	87.8/9.9	86.4/10.6

In the next step, in order to investigate the effects of clustering on the SVM classifier, the selected genes (25 genes) were fed to the SVM classifier, and mean sensitivity, specificity and accuracy were calculated. Table 4.5 gives information on the sensitivity, specificity, and accuracy of the SVM classifier when different clustering methods were used, and the data was partitioned in different amounts of clusters for the leukaemia dataset. It can be seen from Table 4.5 that when data clustering is not applied (1 cluster), sensitivity, specificity, and accuracy were calculated 95%, 97.7%, and 98.1% respectively. An improvement of 0.7% and 0.8% in accuracy was observed when data was clustered into 4 partitions via K-means and C-means respectively compared to the case when no clustering was used. Furthermore, in the case when data was clustered into 3 partitions using C-means, improvement in all performance criteria was achieved compared to when no clustering was utilised. In the case of the prostate cancer dataset, as can be seen from Table 4.6 when clustering was not applied (1 cluster), a higher classification performance was achieved excluding two cases where C-means clustering was used to cluster the data into 3 and 5 partitions.

Table 4.5: SVM classifier performances including the mean sensitivity, specificity, accuracy, and standard deviation (SD) for leukaemia.

		Number of clusters					
		1	2	3	4	5	6
		Mean/SD	Mean/SD	Mean/SD	Mean/SD	Mean/SD	Mean/SD
K-means	Sensitivity	95/5.1	94.9/5.0	94.3/5.7	95/5.1	94.4/5.9	93.5/6.8
	Specificity	97.2/4.1	97/4.2	96.8/4.5	96.8/4.3	96.8/4.5	96.5/4.9
	Accuracy	98.1/3.4	98/3.6	97.1/4.3	98.8/3.1	97.5/4.1	97.7/3.9
C-means	Sensitivity	95/5.1	94.6/5.4	95.5/4.6	95/5.4	94.9/5.1	94.6/5.5
	Specificity	97.2/4.1	97.1/4.1	97.8/3.8	96.7/4.6	96.5/4.8	96.9/4.1
	Accuracy	98.1/3.4	97.8/3.7	98.7/3.1	98.9/2.9	97.2/3.9	97.4/3.8
Hierarchical	Sensitivity	95/5.1	95/5.3	94.8/5.6	94.5/5.9	94.9/5.0	94.6/5.4
	Specificity	97.2/4.1	97.1/4.2	97.2/4.1	96.9/4.3	96.3/4.9	96.8/4.5
	Accuracy	98.1/3.4	97.7/3.7	97.9/3.4	97.7/3.4	98.1/3.2	97.6/4.0

Table 4.6: SVM classifier performances including the mean sensitivity, specificity, accuracy, and standard deviation (SD) for prostate cancer.

		Number of clusters					
		1	2	3	4	5	6
		Mean/SD	Mean/SD	Mean/SD	Mean/SD	Mean/SD	Mean/SD
K-means	Sensitivity	89.4/8.3	88.6/9.1	88.8/9.1	89.1/9.2	89.1/9.2	89.3/8.4
	Specificity	93/6.8	92.8/6.9	92.9/6.9	92.1/7.3	90.5/8.8	92.7/7.2
	Accuracy	90.1/7.6	89.8/7.8	89.6/8.0	89.3/7.9	89.7/7.8	90/7.6
C-means	Sensitivity	89.4/8.3	89/9.1	89.9/8.1	88.6/9.1	89.7/9.0	89.4/9.3
	Specificity	93/6.8	92.6/7.1	93.3/6.5	92.6/7.1	93.1/6.7	92.9/6.9
	Accuracy	90.1/7.6	89.2/8.2	91.4/7.1	89.6/7.9	91.2/7.2	89/8.4
Hierarchical	Sensitivity	89.4/8.3	89.1/8.9	89/9.2	89.1/8.9	88.9/9.3	88.5/9.5
	Specificity	93/6.8	92.8/7.0	93/6.8	92.1/7.3	93/6.3	91.8/7.6
	Accuracy	90.1/7.6	89.8/	89.5/7.8	90.1/7.6	89.7/8.0	88.9/9.2

Figure 4.4 illustrates a comparative performance between SVM and MLP for the leukaemia and prostate cancer datasets when no clustering was applied. It can be seen that the SVM classifier has a better sensitivity, specificity, and accuracy compared to that of the MLP classifier in both datasets.

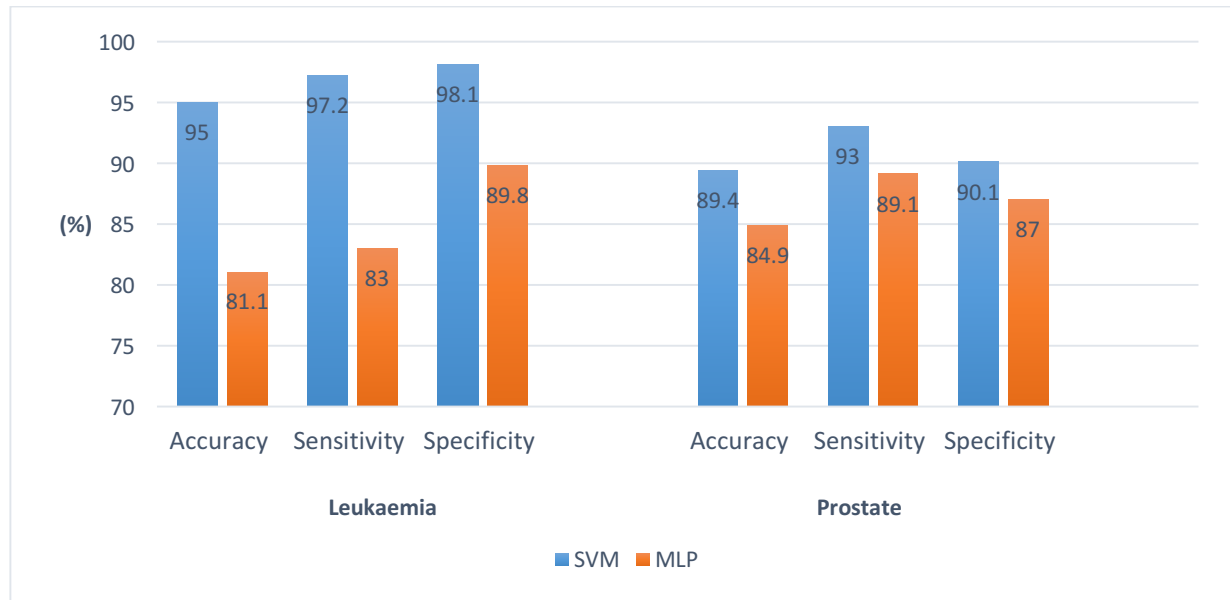


Figure 4.4: MLP vs SVM performance without clustering.

4.6: Proposed Gene Selection Based on Shuffle Technique

4.6.1: Methods

In the previous section, the impact of conventional data clustering on classification performance was investigated, and it was determined that conventional clustering may not have any significant effect on classification performance. In order to fully investigate the effect of data clustering on classification performance, this section investigates the effect of optimisation based clustering methods on the performance of the SVM and MLP classifiers compared to conventional methods. A novel gene selection approach called shuffling is proposed to enhance the selection of the most informative genes

- **Novel shuffle technique to enhance gene selection**

In cancer classification, using clustering based gene selection (grouping genes before gene selection) and changing the number of clusters, both result in selection of different sets of genes (see Equation 4.13). As such, different classification accuracies are obtained. The

differences in the selected genes could occur because the initial centroid protocol for clustering is not specified, but selected randomly [181]. Since the clustering outcome is highly dependent on the initial centroids, it often transpires that better results would have been achieved with other initial points. The standard solution is to try the algorithm a few times with different initial points [182]. As demonstrated in Section 4.5.2, differences in the selected genes can influence the classification performance, so by creating a method to reinforce the selected genes a more robust classification can be performed.

To overcome this problem, a technique called shuffling is proposed in this study. This requires that the data is clustered 6 times, setting different numbers of clusters, ranging from 1 to 6 in each case. As a result, the number of clusters in the first run is set to one, implying no clustering is used, hence the algorithm goes straight to the gene selection step and selects the top 20 genes. In the second run, data is partitioned in two clusters, while the clustering algorithm iterates 100 times to minimise the cost function to achieve clusters that are more accurate. After 100 iterations of the clustering algorithm, gene selection is carried out according to the number of clusters and the population in each cluster. Therefore, depending on the number of clusters, different sets of genes are selected. A similar procedure to the second run carries on until the final run, in which data is clustered into six partitions.

The reason for merging the outputs of a clustering algorithm when the number of clustered varied from 1-6 was to assure reinforcement for the selected genes by the clustering algorithm. To shed light on this let's assume we choose 20 genes when K-mean clustering is used and the number of clustered is equal to two. If we run the K-mean again while the number of clusters are three, a slightly different set of 20 genes will be selected which is due to the nature of selection criteria across different clusters that depends both on the number of genes in the cluster and the ranking of genes in that cluster (see Equation 4.13). If run the algorithm for four clusters, again a different set of 20 genes will be selected. It should be noted that although a different set of genes are selected each time, some genes could be repeatedly selected while changing the number of clusters. Since in the proposed shuffle technique the gene selection is done six times when changing the number of clusters from 1-6, and each time 20 genes are selected, a total 120 genes are acquired. However, some of the 120 genes are similar and repeatedly selected while varying the number of clusters by the same clustering algorithm. These 120 genes are then ranked based on their repetitions of being selected by the clustering algorithm. Therefore, when choosing the 25 most repeated genes we reinforce the selected genes that are chosen by the algorithm and do not rely solely on one outcome of the algorithm. In another word, this introduces a more robust gene selection. The number of final selected genes were chosen 25 to correlate with the number

of selected genes in Section 4.5.1. Furthermore, as it was discussed in Section 4.4.1 the MLP classifier contains 25 input neurons and the 25 selected genes also correlate to this. This was done to ensure the same setup for classifiers in Sections 4.5 and this section (4.6). After selecting 25 most repeated genes, these genes were fed into the MLP and SVM classifiers. Finally, sensitivity, accuracy, and specificity for both SVM and MLP are calculated.

The proposed methodology that incorporates the shuffle technique and the new optimisation algorithm, COA-GA, that was explained in Section 4.2.5, is illustrated in Figure 4.5.

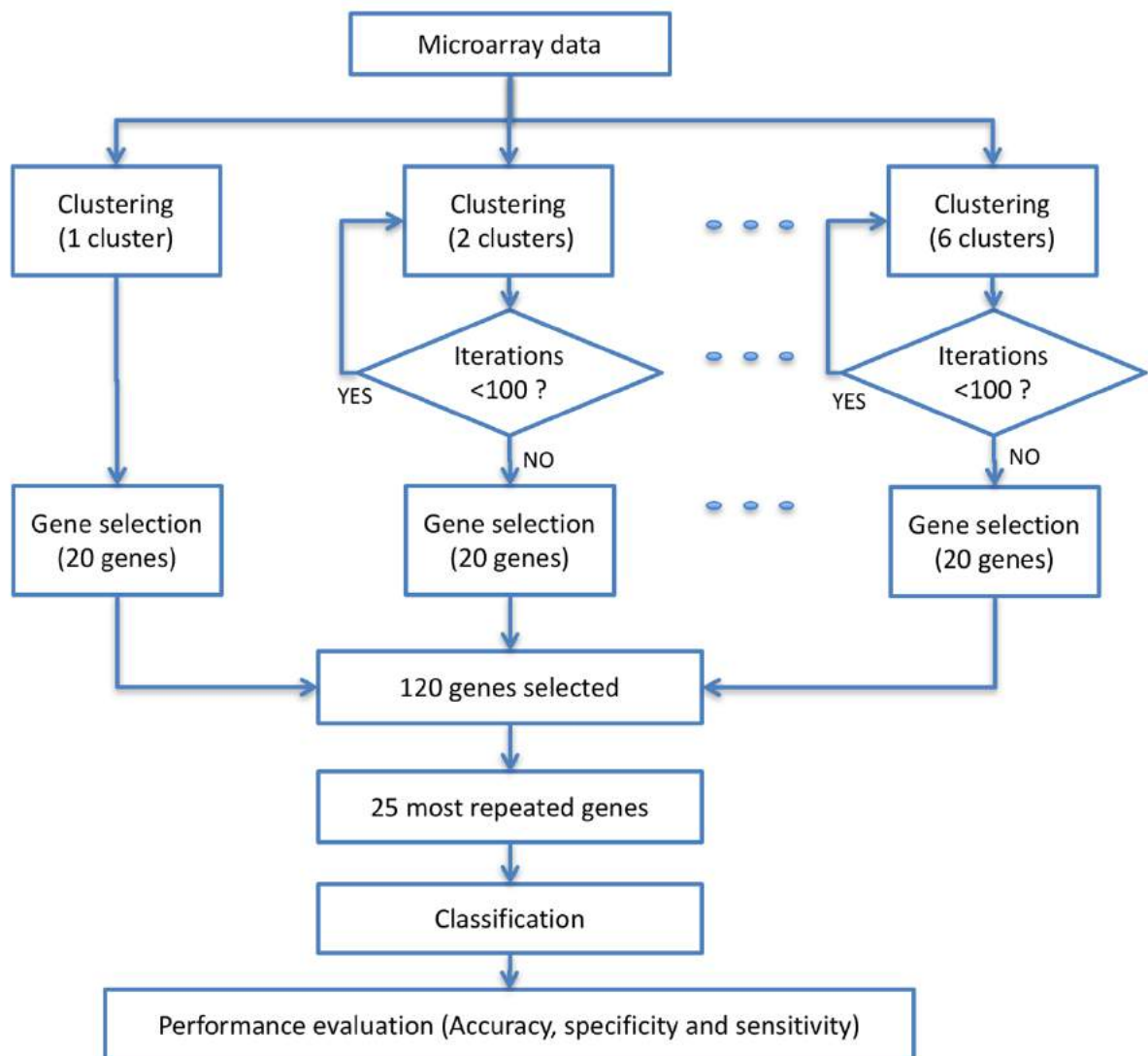


Figure 4.5: Proposed shuffle method.

4.6.2: Results

Basic information relating to the datasets used in this study is listed in Table 4.7, including the number of genes, number of samples, and the two classes for each dataset.

Table 4.7: Basic information of the microarray data used in this study.

Cancer	Genes	Samples	Class1	Class2
Leukaemia	7,129	72	48 (ALL)	25 (AML)
Lymphoma	4,026	47	24(germinal centre B-DLCL)	23 (active B-DLCL)
Prostate	12,600	102	50 (Normal)	52 (Cancerous)

In order to investigate the effects of clustering on the classifier performance, different clustering methods have been used with the shuffle method. These were compared to classification accuracy when no clustering was used, in order to determine the factor providing the greatest increases in classification. In that case, the data was scored based on Equation 4.9, and the 25 genes with the highest scores were extracted for the purpose of classification. Table 4.8, Table 4.9, and Table 4.10 compare how different clustering algorithms with the shuffle method affect the classification accuracy, sensitivity, and specificity of SVM and MLP classifiers for leukaemia, lymphoma, and prostate cancers respectively.

Table 4.8: The mean sensitivity (SE), specificity (SP), accuracy (AC), and standard deviation (SD) of classification results for MLP and SVM classifiers when integrating different clustering algorithms in the shuffle technique for the leukaemia dataset.

Method	MLP			SVM		
	SE/SD	SP/SD	AC/SD	SE/SD	SP/SD	AC/SD
<i>K-means</i>	80.1/16.3	81.0/14.9	86.2/10.8	97.2/3.6	99.8/1.9	98.7/3.2
<i>C-means</i>	82.6/13.6	83.2/13.2	86.7/10.4	96.8/4.3	99.6/2.1	98.8/2.3
<i>Hierarchical</i>	82.8/13.1	82.0/14.2	90.0/7.6	94.6/5.3	96.8/3.8	96.0/4.1
<i>GA</i>	98.2/3.7	87.4/9.1	91.0/6.3	99.0/2.1	99.6/1.9	99.5/2.0
<i>PSO</i>	91.0/7.1	84.2/10.9	90.5/7.8	99.6/1.9	99.4/2.1	99.3/2.2
<i>COA</i>	95.2/5.7	85.8/10.1	93.9/6.2	99.5/1.9	99.5/1.9	99.5/1.9
<i>COA-GA</i>	95.6/5.3	84.9/10.6	93.9/6.6	99.6/2.1	99.9/0.8	99.7/1.1
<i>No cluster</i>	82.0/13.1	83.4/12.9	90.0/7.8	96.0/3.9	98.0/2.5	98.9/2.1

It is noted that in all datasets, the SVM classifier has a higher accuracy, specificity, and sensitivity compared to the MPL classifier. In the case of leukaemia when using the SVM classifier when no clustering was used, sensitivity, specificity, and accuracy were 96%, 98%, and 98.9% respectively, which is comparable to the results when K-means, C-means and hierarchical clustering were used. However, K-means clustering slightly outperforms the C-means and hierarchical clustering by reaching a sensitivity of 97.2% and specificity of 99.8%. Interestingly, in all cases in which optimisation algorithms were used, a higher classification accuracy, sensitivity, and specificity were reached. For both classifiers, COA-GA shows a better performance compared to other optimisation methods. For instance, in the case of SVM, by using COA-GA, sensitivity, specificity, and accuracy reach 99.6%, 99.9%, and 99.7%

respectively.

Table 4.9: The mean sensitivity (SE), specificity (SP), accuracy (AC), and standard deviation (SD) of classification results for MLP and SVM classifiers when integrating different clustering algorithms in the shuffle technique for the lymphoma dataset cancer.

Method	MLP			SVM		
	SE/SD	SP/SD	AC/SD	SE/SD	SP/SD	AC/SD
<i>K-means</i>	86.3/10.2	96.2/4.4	86.4/10.0	90.3/7.5	100/0.0	88.3/8.9
<i>C-means</i>	88.3/8.8	96.7/4.2	87.5/9.1	89.3/7.8	100/0.0	88.5/8.8
<i>Hierarchical</i>	85.9/10.9	97.1/3.9	87.8/8.8	89.6/7.6	100/0.0	88.8/8.7
<i>GA</i>	89.4/7.6	98.7/2.5	90.9/7.3	91.2/6.8	100/0.0	92.4/5.2
<i>PSO</i>	89.1/7.9	98.5/2.8	90.3/7.2	91.8/6.2	100/0.0	92.9/4.9
<i>COA</i>	91.2/6.9	98.9/2.2	91.6/6.8	92.3/5.3	100/0.0	93.1/4.2
<i>COA-GA</i>	92.1/6.1	99.5/1.8	92/5.9	93.2/4.9	100/0.0	93.9/3.8
<i>No cluster</i>	87.4/9.2	96.6/3.9	87.7/9.1	90.1/7.9	100/0.0	88.9/8.6

Similar trends appear across all three data sets, whereby optimisation based clustering yields a better classification accuracy, sensitivity, and specificity. For instance, in the case of lymphoma when SVM is used and COA-GA is applied for clustering data, an accuracy of 93.9% is achieved, compared to 88.9% when no clustering is used (see Table 4.9). For the same dataset, it can be seen that a specificity of 100% is achieved with all algorithms for the SVM classifier. For prostate cancer, as presented in Table 4.10, the classification accuracy, sensitivity, and specificity when utilising traditional clustering methods are comparable to the case in which no clustering is used. In contrast, an improvement of 5.1% is seen for SVM accuracy, and 5% for MLP accuracy when COA-GA is used compared to when no clustering is applied.

Table 4.10: The mean sensitivity (SE), specificity (SP), accuracy (AC), and standard deviation (SD) of classification results for MLP and SVM classifiers when integrating different clustering algorithms in the shuffle technique for the prostate dataset cancer.

Method	MLP			SVM		
	SE/SD	SP/SD	AC/SD	SE/SD	SP/SD	AC/SD
<i>K-means</i>	86.1/10.3	90.7/7.1	89.9/7.9	90.1/7.6	94.3/4.5	91.4/6.4
<i>C-means</i>	84.2/11.1	90.9/6.9	89.6/8.1	90.9/7.1	94.6/4.3	91.2/6.8
<i>Hierarchical</i>	86.0/10.3	90.3/7.4	89.0/8.8	91.0/6.8	93.3/4.9	90.4/7.3
<i>GA</i>	90.8/7.1	92.1/5.9	92.1/5.9	95.9/4.2	96.0/3.8	94.3/4.6
<i>PSO</i>	90.5/7.3	92.9/5.3	92.8/5.4	94.6/4.4	96.8/3.7	94.4/4.5
<i>COA</i>	91.5/6.5	93.1/4.9	93.4/4.7	96.1/3.9	98.6/2.8	95.9/4.1
<i>COA-GA</i>	92.4/5.9	94.6/4.1	94.2/4.3	96.9/3.6	99.5/2.1	96.6/3.8
<i>No cluster</i>	86.0/10.4	90.7/7.2	89.2/8.4	90.4/7.5	94.5/4.4	91.5/6.3

Figure 4.6 and Figure 4.7 illustrate the differences in classification accuracy, sensitivity, and specificity when no clustering is used compared to COA-GA clustering for the MLP and SVM classifiers respectively. It can be seen that SVM outperforms the MLP classifier in all cases. Furthermore, the performance is enhanced for both classifiers when the shuffle method integrated with COA-GA is used, compared to that when no clustering is applied prior to gene selection. These results suggest that the shuffle technique with the proposed algorithm (COA-GA) can improve cancer classification performance, and better results could be achieved if SVM is used compared to those if MLP was used as the classifier.

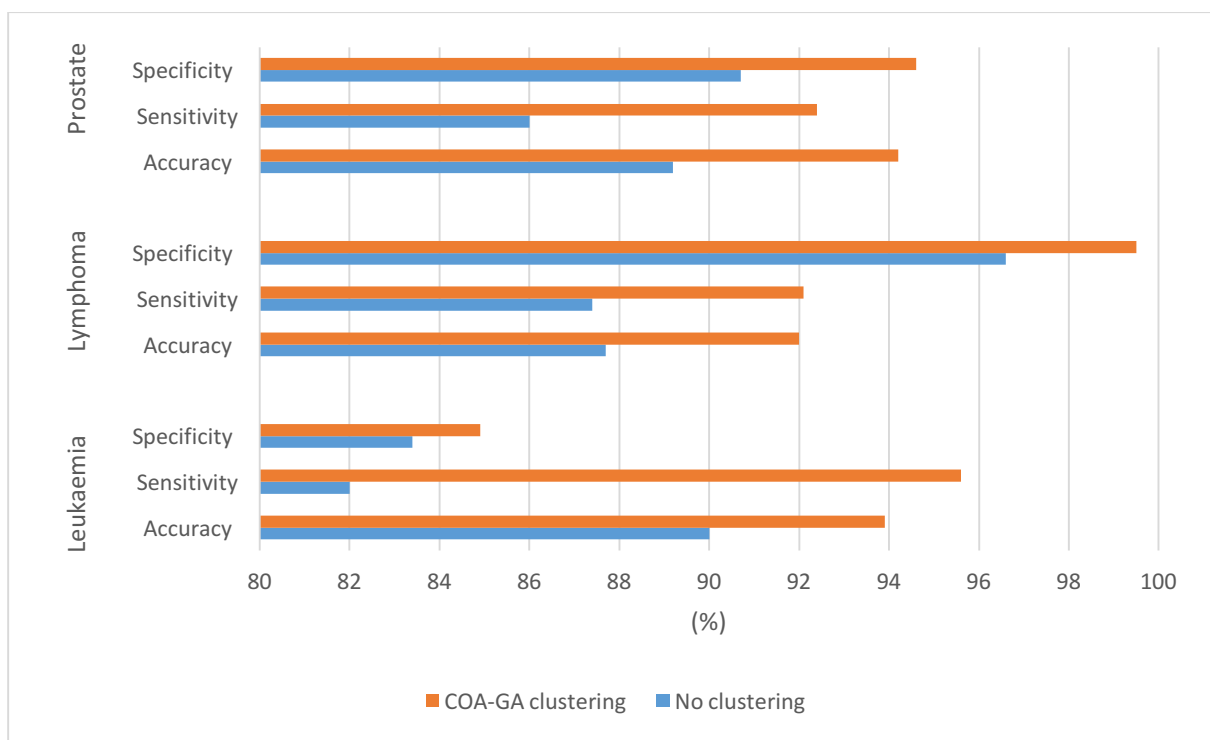


Figure 4.6: Accuracy and sensitivity of MLP classifier results for three cancer datasets when no clustering is used, compared to using the shuffle technique with COA-GA for clustering.

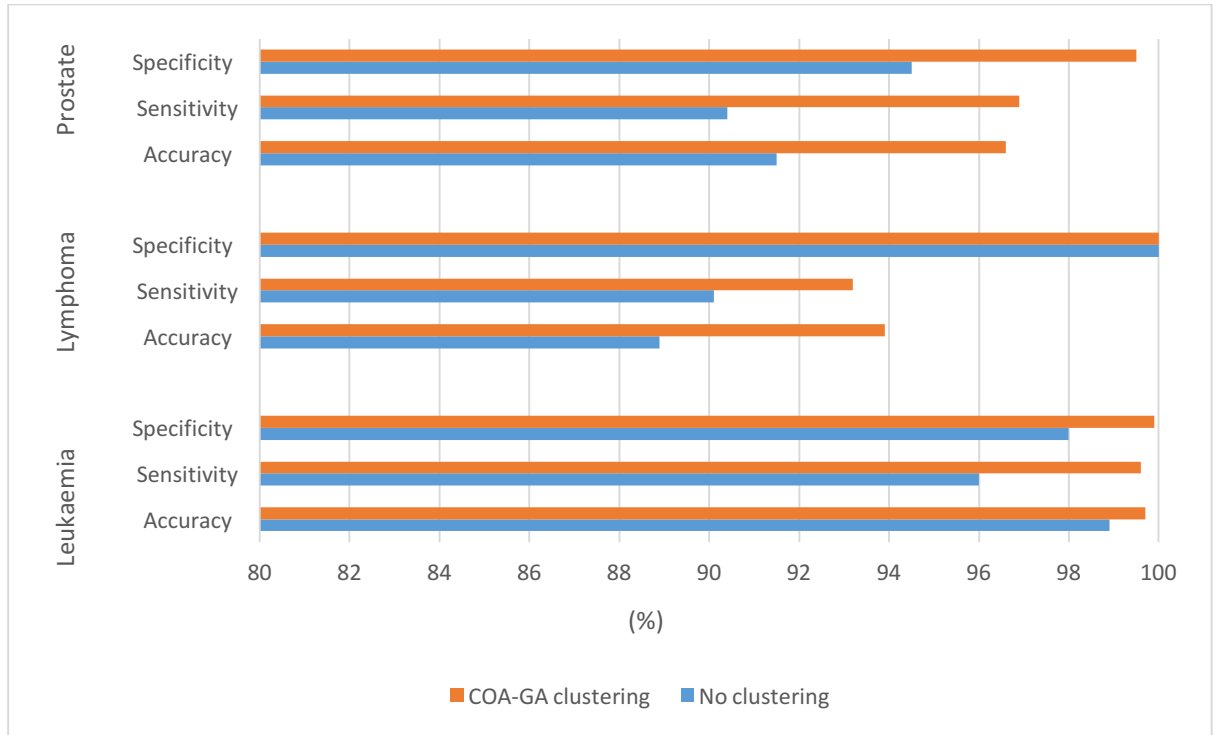


Figure 4.7: Accuracy and sensitivity of SVM classifier results for three cancer datasets when no clustering is used, compared to using the shuffle technique with COA-GA for clustering.

Figure 4.8, Figure 4.9, and Figure 4.10 show the cost function minimisation for COA-GA, COA, GA, and PSO over 100 iterations for leukaemia, lymphoma, and prostate cancer respectively.

For the leukaemia dataset, GA has reached its minimum by the 91st iteration, and the best-cost value reached is 1.500×10^6 . By contrast, PSO has reached its minimum by the 85th iteration, and the best-cost value reached is 1.943×10^6 , indicating that GA has outperformed PSO as it further minimises the cost function. However, the COA algorithm reaches its minimum 2.429×10^5 at 13 iterations, whilst COA-GA could minimise the cost value to 2.400×10^5 at 13 iterations. It can be seen that all four methods reached their minimum after some iterations. However, COA and COA-GA notably outperform GA and PSO, finding a better minimum for the cost function, as well as having faster convergence, whilst COA-GA performed the best among the four algorithms.

A very similar trend in cost minimisation capabilities for all algorithms is observed for the cases of and lymphoma (Figure 4.9). It is notable that in the lymphoma dataset, the COA-GA algorithm significantly outperforms other algorithms, and when compared to COA performance, COA-GA continues to minimise the cost function after the 11th iteration, while COA reached its minimum at this iteration. In the case of prostate cancer (Figure 4.10), it can be seen that COA-GA and COA show the best performance, where COA-GA outperforms COA.

However, it is clear that in this case, PSO significantly outperforms GA opposite to the case of leukaemia and lymphoma where GA outperformed PSO.

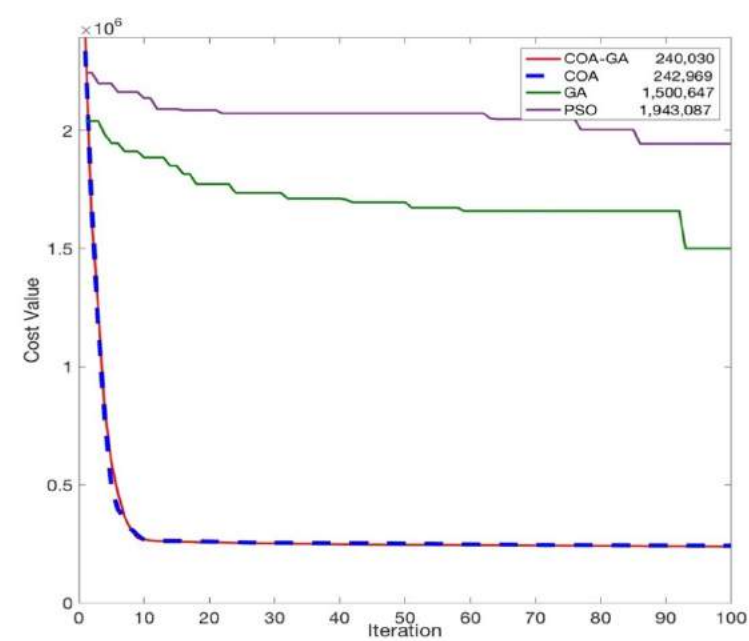


Figure 4.8: Cost minimisation for four algorithms over 100 iterations for leukaemia.

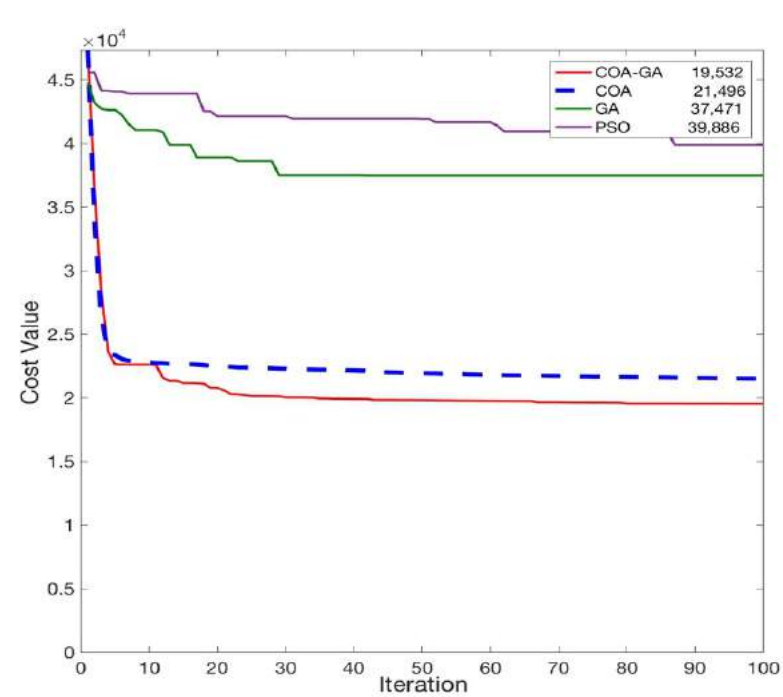


Figure 4.9: Cost minimisation for four algorithms over 100 iterations for lymphoma.

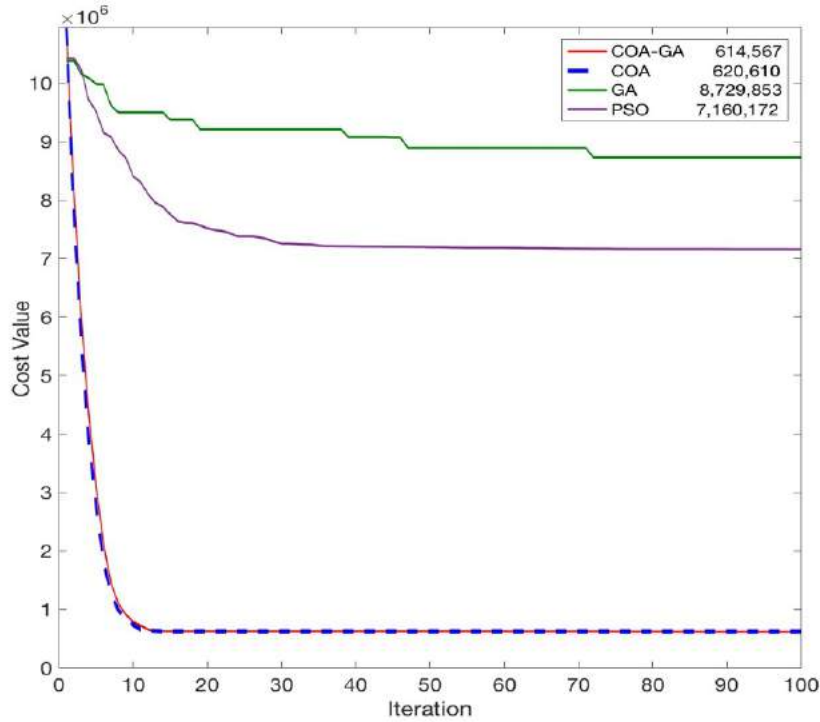


Figure 4.10: Cost minimisation for four algorithms over 100 iterations for prostate cancer.

4.7: Summary

In this chapter, the effects of data clustering prior to gene selection on classification performance was investigated. To this end, first the effects of conventional data clustering methods on classification performance for cancer datasets were investigated. This approach included three steps (i) clustering, (ii) gene selection, and (iii) classification. Three different methods, K-means, fuzzy C-means, and hierarchical clustering; and two classification methods, support vector machine (SVM) and multi-layered perceptron (MLP) neural networks were studied. The results obtained suggest that conventional clustering methods may not impact the classifier performance. This has been observed in the case of both classifiers. The results also suggest that the performance of the SVM classifier is better than that of the MLP artificial neural networks.

In the next step, in order to fully examine the effect of data clustering on classification performance, the effect of optimisation based clustering algorithms on the performance of the SVM and MLP classifiers were investigated and compared to conventional methods. A novel approach to enhance gene selection called the shuffle technique was proposed, in which a new hybrid algorithm, COA-GA, was implemented for clustering microarray data. The performance of the proposed algorithm was tested against other well-known optimisation

algorithms including PSO, GA, and COA. The results suggested that data clustering with optimisation based clustering methods prior to gene selection via the proposed method significantly enhance the performance of both classifiers. However, clustering data via conventional clustering methods did not have any impact on any of the classifiers' performances that were used in this investigation. It was also explained that when no clustering was used, the results were comparable with the cases where conventional clustering methods were used. Comparative analysis between the proposed hybrid algorithm, COA-GA, with other optimisation algorithms like PSO, GA, and COA, suggested that COA-GA significantly outperforms other algorithms at reaching a better minimum in fewer iterations. In the final part of this chapter, better classification performance was achieved when SVM was used compared to when the MLP classifier was used for all cancer datasets.

Chapter 5: Two Stage Gene Selection for Cancer Classification Using Microarray Data

5.1: Introduction

It is now well established that early diagnosis of tumours can greatly increase the rate of cancer survival by providing the right treatment at early stages. However, methods such as X-ray imaging and computed tomography (CT) usually detect such tumours in later stages of cancer formation. Nevertheless, invasive methods such as surgery could detect malignancies, with the downside of potential severe side effects, and therefore such methods are not recommended for benign cases [183]. In this respect, over the last few decades gene expression profiling using microarray technology has attracted many scientists towards the early detection and classification of cancer [184,185].

However, as discussed in Chapters 1 and 2, due to the so-called 'curse of dimensionality' problem, the prognosis and classification tasks remain challenging to date. High classification accuracy is of the utmost importance for personalised medicine. Since there are two important factors that can enhance the classification performance, gene selection and classifier method, computer scientists have proposed different methods to increase the efficiency of each factor.

With regards to the gene selection factor, numerous studies have been carried out with the objective of increasing the classification accuracy [146,186]. For example, Golub *et al.*, [26] proposed a signal-to-noise ratio method, which was also used later in different studies [187]. Cho *et al.* [188] investigated several methods such as Pearson's correlation, Euclidean distance, information gain, and mutual information to select the most informative genes

among different cancer types including colon, lymphoma, and leukaemia. In the above methods, genes are first ranked based on the relevant criterion, and then the top n genes are selected for classification purpose. Several classification methods have been proposed for such analysis, and LDA, k-NN, SVM, and MLP artificial neural networks were discussed in Section 3.6. Most of these gene selection approaches, known as filter methods, have greatly contributed to early detection and classification of cancer by providing useful information for medical experts. Nevertheless, the downside of the filter methods, which is ignoring feature dependencies and interaction with classifiers, can lead to poor classification accuracy.

To address this problem, evolutionary algorithms such as GA and PSO have been applied for the purpose of gene selection. These methods essentially are heuristic optimisation algorithms that find the optimum subset of features to achieve the best classification accuracy, which is feasible as these methods are combined with the classification step in the form of a hybrid setup. For instance, Lee et al. proposed a gene selection method using an adaptive genetic algorithm combined with a KNN classifier to achieve a good classification accuracy for colon cancer datasets [189]. Shen *et al.* proposed a method combining discrete PSO and SVM for the selection of the most informative genes. The result of this study suggested that the SVM performance was significantly enhanced when PSO is used (91.7%) compared to the case when no gene selection was applied (83%) for a colon cancer dataset [179]. Since finding the local optimum is challenging for most optimisation algorithms, some studies proposed hybrid optimisation methods to overcome this problem. For example, Li et al. proposed a hybrid method combining PSO and GA that used SVM as the classifier [154]. This method was applied to different cancer datasets, and the result suggested that their proposed method can select the most informative genes that enhance classification accuracy.

The vast majority of these studies focus on increasing the classification accuracy rather than the number of selected genes. Biomarker identification is another area of ongoing research, where it is important to identify a small number of genes in order to spot patterns. For instance, choosing a few genes that are all differentially expressed across different samples [190,191]. Works of research argue that the ideal classification task should result in the highest classification accuracy with less genes [192]. Therefore, it is essential to create a model for cancer classification that meets both objectives for tumour classification. To date, it has been possible to achieve the highest classification for some cancer datasets. However, even in these cases, several genes are needed to be used to achieve the highest classification. Therefore, in this chapter, the main objectives are to select the optimum number of most informative genes that can best distinguish between two cancer types to achieve the highest classification accuracy. To accomplish these objectives, a new optimisation algorithm which

combines the cuckoo optimisation algorithm (COA) and harmony search (HS), is proposed (COA-HS), which will be used in a two-stage gene selection method.

5.2: Proposed Method

The general methodology used in this study is illustrated in Figure 5.1. First, the data was discretised into nine states. After this pre-processing stage, the top 100 genes which are the most relevant and least redundant were selected using the minimum redundancy and maximum relevance (MRMR) feature selection (a filter method [193]). The selected genes were fed to a wrapper setup that consisted of the COA-HS algorithm and SVM classifier, to choose the minimum number of genes that provide 100% accuracy. Using two-stage gene selection combines the advantages of both filter and wrapper methods of gene selection. Finally, the classification performance for the selected genes was measured in terms of accuracy via the leave-one-out cross validation method (LOOCV). In order to validate the performance of COA-HS, the results were compared to those established with other evolutionary algorithms, such as the genetic algorithm (GA), particle swarm optimisation algorithm (PSO), harmony search algorithm (HS), and cuckoo optimisation algorithm (COA).

Microarray data for three cancer types (leukaemia, prostate, and lymphoma) was used in this study. Gene expression data for leukaemia [26] and prostate cancer [30] was obtained from the Broad Institute (www.broadinstitute.org); Gene expression data for lymphoma [28] was obtained from the Lymphoma/Leukaemia Molecular Profiling Project (llmpp.nih.gov). Basic information relating to the datasets used in this study is provided in Table 5.1, including the number of genes, the number of samples and the two classes for each dataset.

Table 5.1: Basic information of the microarray data used in this study.

Microarray dataset	Number of genes	Number of samples	Class1	Class2
Leukaemia	7,129	72	47 (ALL)	25 (AML)
Prostate	12,600	102	50 (Normal)	52 (Cancerous)
Lymphoma	4,026	47	24 (Germinal centre B-DLCL)	23 (Active B-DLCL)

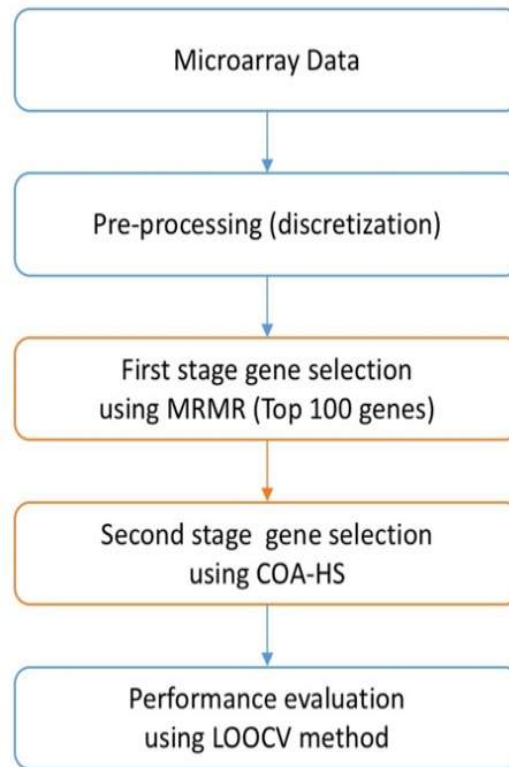


Figure 5.1: Schematic of the general methodology for gene selection.

5.3: Discretisation of Data

Discretisation is the process of converting continuous values into discrete counterparts, and this technique is frequently used as a pre-processing step in the analysis of biological data for several reasons. For instance, some gene selection and classification methods only accept discrete values as their input. Although the representation of data is changed through this process, it is assumed that the biological information within the data is preserved. In fact, several studies suggest that using discrete values can lead to more efficient learning processes [194–196]. Furthermore, Peng et al. investigated the performance of continuous and discrete values of microarray data in classification performance, and suggested that discrete values lead to a better classification performance [193].

In the context of microarray gene expression data, there are several discretisation methods that can be applied, which can be categorised into supervised and unsupervised methods. In the supervised method, gene expression data is discretised while taking into consideration the class information of each gene (healthy vs cancerous). In contrast, in unsupervised methods, gene expression values are discretised without any impact from their class label. In this chapter, the unsupervised approach is the focus. In unsupervised cases, there exists two

pathways. One method is discretising data based on absolute values of gene expression, and another is discretising based on variation between time points [197,198].

In this chapter, data is discretised using absolute values of gene expression in order to reduce the noise in the gene expression data, and to enhance the accuracy of classification results [199]. Gene expression values for each gene were categorised into a nine-state variable based on the mean value (μ) and standard deviation (σ) for that gene. For each gene, the nine states showed whether the gene was not expressed (state zero) or if expressed, how much it was over-expressed (states +1 to +4) or under-expressed (states -1 to -4). Table 5.2 details the different states utilised in the discretisation of data.

Table 5.2: Discretisation of gene expression data.

Data	States	Data	States
$\mu < d < \mu - 1/2 \sigma$	0	$\mu < d < \mu + 1/2 \sigma$	0
$\mu - 1/2 \sigma < d < \mu - \sigma$	-1	$\mu + 1/2 \sigma < d < \mu + \sigma$	1
$\mu - \sigma < d < \mu - 3/2 \sigma$	-2	$\mu + \sigma < d < \mu + 3/2 \sigma$	2
$\mu - 3/2 \sigma < d < \mu - 2\sigma$	-3	$\mu + 3/2 \sigma < d < \mu + 2\sigma$	3
$d < \mu - 2\sigma$	-4	$d > \mu + 2\sigma$	4

As mentioned earlier, Peng *et al.* (2005) concluded that discrete values lead to a better classification performance for microarray data. A study by Gallo *et al.*, (2015) suggests that although the number of states in a discretisation task depends on the inference of the algorithm that the data is prepared for, there is a trade-off between computational complexity and the loss of information when choosing the number of states. On the one hand, by increasing the number of states one can better preserve the information. On the other hand, by increasing the number of states the computational complexity also significantly increases [198]. In this study data was discretised into nine states. To visualise the effects of discretisation, the frequency plots before and after discretisation for lymphoma (see Figure 5.2), prostate (see Figure 5.3), and leukaemia (see Figure 5.4) cancer datasets are provided.

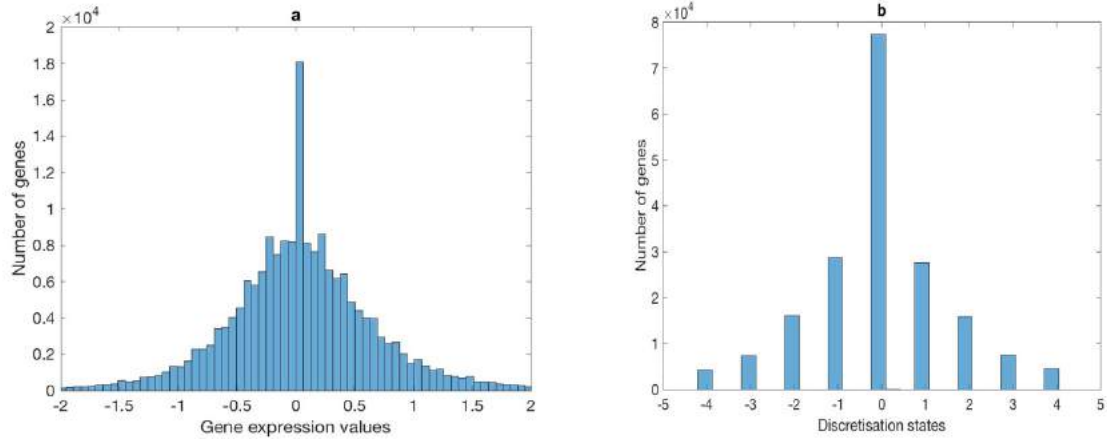


Figure 5.2: Frequency plots before (a) and after (b) discretisation for lymphoma dataset.

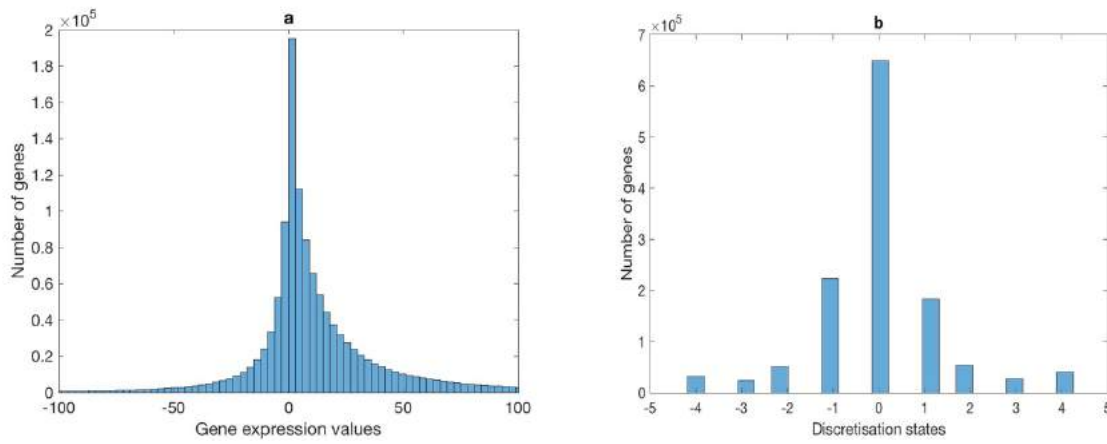


Figure 5.3: Frequency plots before (a) and after (b) discretisation for prostate dataset.

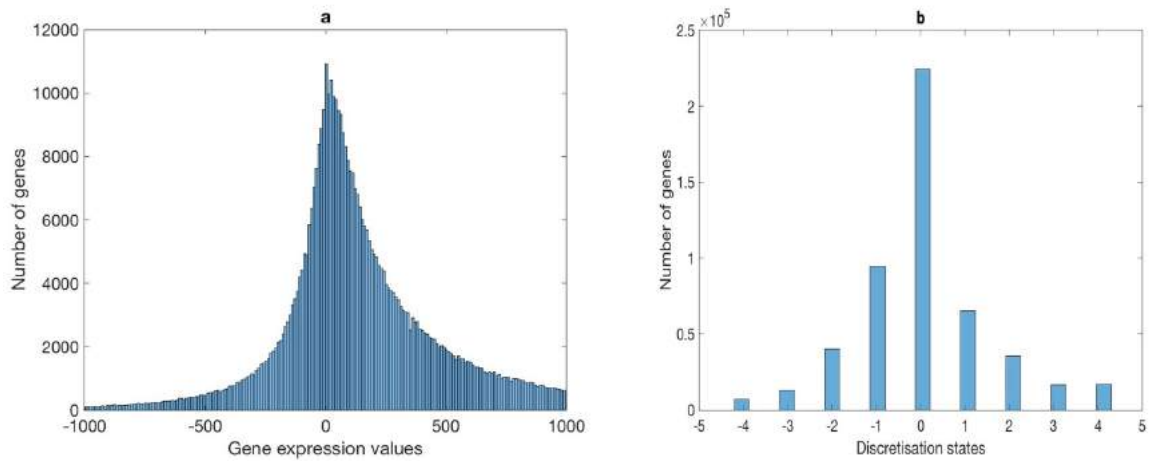


Figure 5.4: Frequency plots before (a) and after (b) discretisation for leukaemia dataset.

5.4: First Stage Gene Selection Using Minimum Redundancy Maximum Relevance (MRMR)

The goal of feature selection in a classification task is to identify a subset of features that best characterise the statistical significance of the classification task [46]. Since utilising wrapper methods for gene selections are computationally expensive when dealing with gene expression data due to existence of thousands of genes, filter methods are usually used for gene selection or applied to reduce the dimension of the data before applying a wrapper technique. Feature entropy is an appropriate metric to identify such informative genes. Entropy refers to the initial uncertainty of the output class [200], and can be calculated using Equation 4.1.

$$H(A) = - \sum_{a=1}^{N_a} P_a(a) \log(P_a(a)) \quad (4.1)$$

where $\{P_a(a) \mid a = 1, 2, \dots, N_a\}$ is the probability density for different classes. A conditional entropy is used to define the mean uncertainty with respect to the feature vector, which can be calculated via following expression:

$$H(A|B) = \sum_{b=1}^{N_b} P(b) \left(\sum_{a=1}^{N_a} P_a(a|b) \log(P_a(a|b)) \right) \quad (4.2)$$

where b is the input feature vector with N_b samples and $P_a(a|b)$ is the conditional probability of class a from feature vector b . Initial entropy is usually larger than conditional entropy; however, in the case of total independence between the output class and feature, both entropies have equal values. Mutual information that quantifies the mutual dependencies of two variables A and B can be defined based on Equation 4.3.

$$I(A; B) = H(A) - H(A|B) \quad (4.3)$$

This equation can be rewritten as:

$$I(A; B) = \sum_{a \in A} \sum_{b \in B} p(a; b) \log \frac{p(a; b)}{p(a)p(b)} \quad (4.4)$$

where $p(a)$ and $p(b)$ are the probability density functions of variables A and B respectively, and $p(a; b)$ is the combined probability density function of both variables. Mutual information between two variables A and B defines how much information about variable B one can gain by only looking at variable A .

Minimum redundancy maximum relevance feature selection (MRMR), which is a filter method, uses mutual information to select those genes that were mutually maximally dissimilar, but with the highest relevance to the target class [193]. First, in order to choose a subset of genes that best represents the entire dataset, the minimum redundancy was calculated using the following equation.

$$\min W, W = \frac{1}{|T|^2} \sum_{g_i, g_j \in T} I(g_i; g_j) \quad (4.5)$$

where T denotes the total number of important genes that were required to be extracted, and $I(g_i; g_j)$ represents the mutual information of gene i and gene j . Next, the mutual information between genes (g_i) and the corresponding classes (C), $I(g_i; C)$ were calculated to quantify the relevancy of each gene with regards to its class. Subsequently the maximum relevancy was acquired using Equation 4.6. Maximum relevancy selected the top T genes in the descending order of $I(g_i; C)$ [193].

$$\max V, V = \frac{1}{|T|} \sum_{g_i \in T} I(g_i; C) \quad (4.6)$$

Since both conditions W and V were equally important, MRMR combines both. This combination could be carried out by two methods, namely MRMR_{MIQ} and MRMR_{MID} , which combine both conditions as Equation 4.7 and Equation 4.8 respectively. In this study, MRMR_{MIQ} was used, which is formulated as Equation 4.9.

$$\max(V - W) \quad (4.7)$$

$$\max\left(\frac{V}{W}\right) \quad (4.8)$$

$$\text{MRMR}_{\text{MIQ}} = \max_{i \in \Omega_T} \left(\frac{I(g_i; C)}{\frac{1}{|T|} \sum_{g_i \in T} I(g_i; g_j)} \right) \quad (4.9)$$

In the proposed method for the first stage selection, by using Equation 4.9, the top 100 genes which were mutually maximally dissimilar were extracted and fed to the second stage

of selection, which used an evolutionary algorithm to select the minimum number of genes that gives the maximum accuracy for the SVM classifier.

5.5: Second Stage Selection Using Evolutionary Algorithms

Fundamentally, optimisation is the process of finding the best solution among all possible solutions. Population based optimisation algorithms initially choose a random set of solutions (initial population), and this population is enhanced via an iterative process. For each iteration, a cost function is established to quantify the outcome of the optimisation task. Since the problem in this study is defined as classification of microarray data while achieving higher accuracy through the minimum number of selected genes, the cost function is designed as follows.

$$\text{Cost function} = \frac{a}{\text{Accuracy}} + \text{NOG} \quad (4.10)$$

where *Accuracy* is the accuracy of SVM classifier measured by the LOOCV method and ranges from 0 to 1, *NOG* is the number of selected genes which ranges from 1 to 100, and *a* is a coefficient. Since the accuracy was more important than the number of selected genes, *a* was set to 1000 in order to give the accuracy more weight in the cost function. Therefore, by minimising the cost function, the number of selected genes is minimised while the accuracy is maximised.

It should be noted that both terms in Equation 5.10 are important for the cost function. For instance, if an algorithm selects 20 genes and gives 0.98 for accuracy, this would result in a cost value of 1040.4. If in another case this algorithm chooses 8 genes and give 0.97 for accuracy, this would result in a cost value of 1038.99. In this scenario, the latter would be more preferable for the algorithm although the accuracy of the preferred case is less. However, if scenario changes such that 0.98 accuracy is acquired by 17 genes, which leads to a cost value of 1037.4, then the algorithm prefers this option than the case of 0.97 with 8 genes. The value for *a* was chosen 1000 as it was observed that this will result in better outcomes. For instance, if *a* was chosen 100 and 35 genes were selected by an algorithm that resulted in 0.9 accuracy this would give a cost value of 146.1. If 3 genes were selected and the accuracy for the 3 genes was 0.7, this results in a cost value of 145. In this scenario, the algorithm would prefer the latter case although in the latter case the accuracy is only 0.7. as a result, the algorithm might select less number of genes, but it leads to a poor classification performance.

In respect to LOOCV method, one sample is treated as a test sample, whilst the remaining samples are used for training the SVM, and the accuracy is calculated. If there are N samples, this procedure is repeated N times, each time with a different sample, and the average accuracy is calculated for the selected genes. SVM was used for the classification of selected genes, as the SVM classifier is a powerful classification algorithm and has been demonstrated to exhibit excellent performance in a variety of biological classification tasks [201]. The LOOCV method was chosen as it can overcome data overfitting [202].

A new hybrid optimisation algorithm, COA-HS, was developed by combining the recently invented COA [24] and HS algorithms. The results were compared with the GA, PSO, COA, and HS algorithms. Details of GA, PSO, and COA algorithm can be seen in Section 4.2.2, Section 4.2.3, and Section 4.2.4 respectively. Details of HS and the proposed COA-HS will be described in the following subsections.

5.5.1: Harmony Search Algorithm (HS)

The harmony search (HS) is a musically-inspired optimisation algorithm [203]. In jazz, musicians improvise their instruments' pitch in order to find a perfect harmony, which can be achieved through three options. The first option is to play a pitch from memory. The second option is to play a random pitch within the acceptable range of available pitches. Finally, they can play a pitch adjacent to a pitch in their memory. In the HS algorithm, these options are respectively referred to as harmony memory (HM), pitch adjustment rate (PAR) and harmony memory consideration rate (HMCR). Figure 5.5 illustrates the analogy of the musical improvisation process and optimisation process.

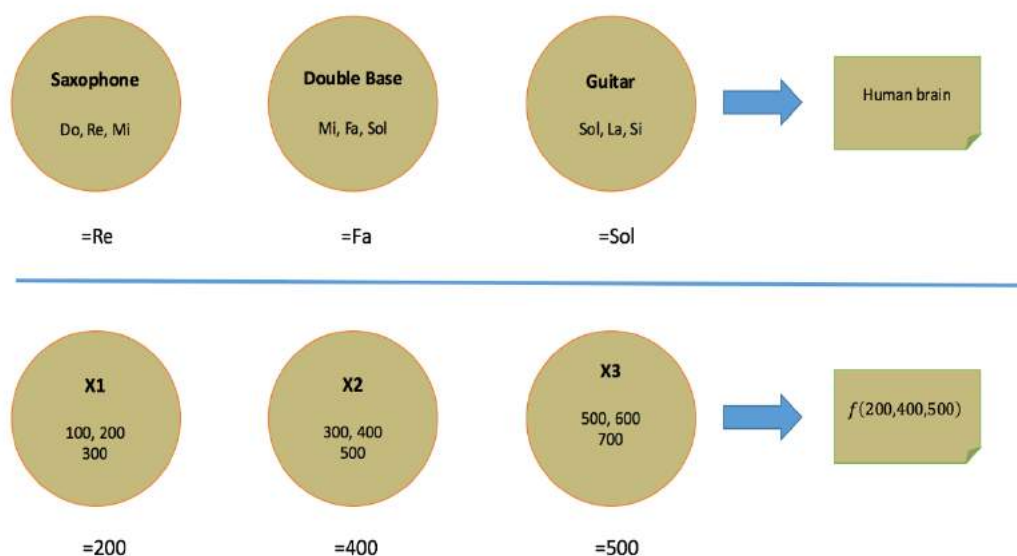


Figure 5.5: Analogy between musical improvisation process and optimisation process.

The HS algorithm has been successfully applied to various optimisation problems, such as feature selection [204], discrete design variables [205], and continuous optimisation problems [206].

The harmony search algorithm follows a number of steps as demonstrated below. The cost minimisation plot is acquired to visualise how HS minimises the cost function over 100 iterations.

1. Initialise HMCR and PAR.
2. Initialise harmony memory (HM).
3. Improvise a new harmony memory.
4. Update harmony memory (HM).
5. If the number of iterations is less than 100, go to step 3.
6. Save the best harmony memory as the 'best answer'.

It is noted that HMCR and PAR values affect the performance of the HS algorithm. For instance, HMCR is important in the convergence of the algorithm as this parameter is used to warrant the best fitted solutions are considered as the features of new solutions. The value for this parameter ranges between 0 to 1. It is recommended to choose a value in a range of [0.70 - 0.9] to make ensure enough exploitation [207,208]. This parameter act as crossover rate in genetic algorithm. For instance, if its value is 0.8 this means there is 80 % probability that the value of variables in HM will be chosen for new solutions. PAR is also very important parameter in improvisation process and act like the mutation parameter in the genetic algorithm. This parameter defines whether the variables of the new solutions should be altered to the value of its neighbour variable. PAR value also ranges from 0 to 1, and determines the probability of changing the variable values. The recommended range of values for PAR is [0.1 - 0.3] [208,209]. In order to ensure the optimum values for HMCR and PAR are selected for the cost function and related datasets, the recommended ranges for HMCR and PAR were investigated. To this end the HS algorithm was ran 100 iterations and the final cost value was obtained for leukaemia, prostate, and lymphoma datasets (See Table 5.3). It was observed that the optimum values for HMCR and PAR across all datasets were 0.9 and 0.3 respectively and therefore these values were used for this research.

Table 5.3: Results of using HS with different PAR and HMCR values.

HMCR	PAR	Leukaemia	Prostate	Lymphoma
0.7	0.1	1031	1054	1028
	0.2	1028	1053	1027
	0.3	1030	1052	1027
0.8	0.1	1029	1055	1032
	0.2	1031	1057	1030
	0.3	1032	1052	1029
0.9	0.1	1027	1056	1027
	0.2	1022	1050	1027
	0.3	1021	1045	1023

5.5.2: Proposed Algorithm COA-HS

For this study, a new algorithm was developed by combining the COA and HS algorithms (see Figure 5.6). As discussed, in the COA algorithm (Section 4.5.3), each egg in a nest represents a solution, and each cuckoo represents a new solution. Therefore, in the analysis of gene expression data, a solution refers to a gene. The COA-HS algorithm starts with the initialisation of the cuckoos. After the initial population lay eggs, the profit values of the eggs are calculated by evaluating the cost function. These solutions (eggs) are then fed to the HS algorithm in order to explore more solutions. These can be provided by the improvisation process through HMCR and PAR, which were set to 0.9 and 0.3 respectively. As a result, a better solution can be achieved by preventing premature convergence of the COA.

After the HS algorithm stops, the profit value for the solutions suggested by the HR are calculated through the cost function. Then the profit values of the solutions suggested by the COA and HS are compared, and the solution (egg) with the higher profit value is chosen to survive. Afterwards, these eggs grow and become cuckoos and the survival rate of each cuckoo is calculated. Then all cuckoos move towards the nest with the highest survival rate and lay eggs within the ELR of the best nest (best position). This means that the space of solutions is refined towards the best solution, concluding one iteration of the COA-HS algorithm. This process is repeated 100 times, and each time the cuckoos lay eggs in a further improved position, which results in finding a better solution based on evaluating the cost function.

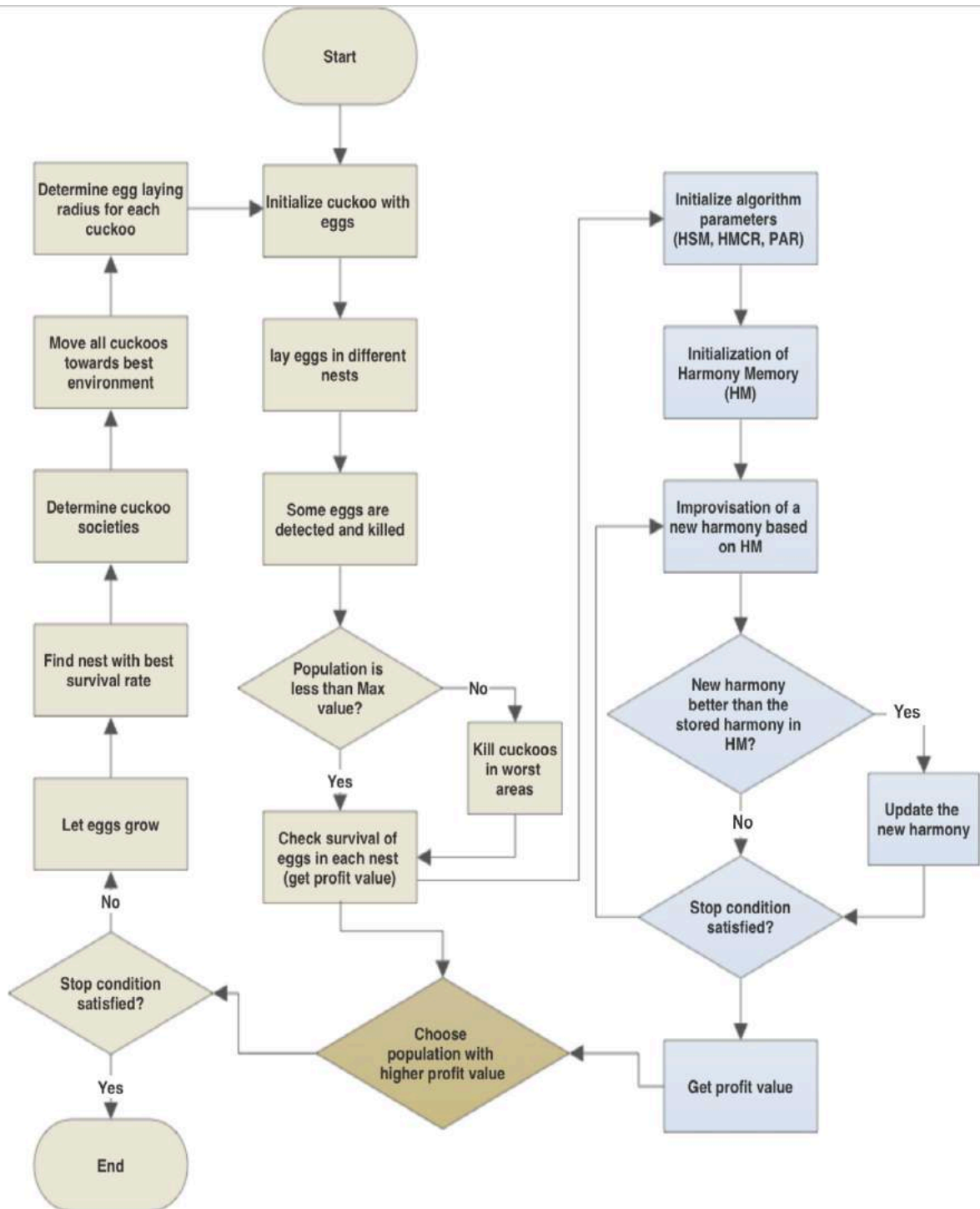


Figure 5.6: Flowchart of COA-HS.

5.6: Results

To select the minimum number of genes that can best distinguish between two classes of cancer, first the number of candidate genes was reduced to 100 using MRMR. These 100 genes were then fed to our proposed algorithm COA-HS to select the best genes while maintaining the highest accuracy. The SVM classifier was used for classification where the Gaussian kernel was employed. The accuracy of the SVM classifier was measured after cross validating using the LOOCV method.

In order to account for possible overfitting, the gene expression samples in each dataset were split into two sets of 25% and 75%. In the splitting task the ratio of class I and class II data (each dataset had two classes) was considered to ensure in each set both classes are presented. The set with 75 % of data was used for training and cross validating the classifier model. Once an optimisation algorithm select the best genes based on the cost function in the second stage of gene selection, the trained model that was used to select the final genes was then used to classify unseen data where the set of 25% of data was used.

Figure 5.7 illustrates the accuracy of the SVM classifier for the top 100 genes selected via MRMR. Initially, as the number of genes increases up to 5-6 genes, the accuracy increases. After initial increase, in some instances the classification accuracy was reduced as the number of genes increased. For example, in the case of the prostate cancer dataset, the classification accuracy for the first 8 genes is 97%, but the accuracy reduces as the number of genes increases, attaining values of 91-93% when 90-100 genes are used.

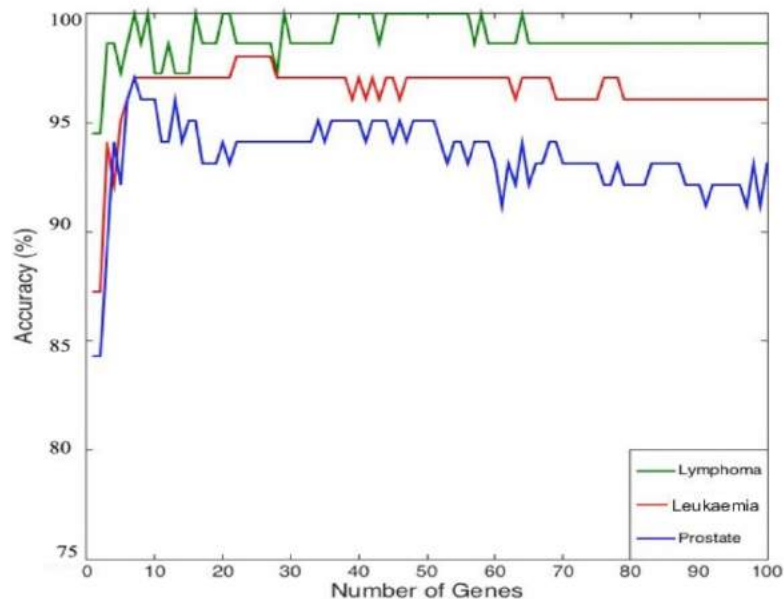


Figure 5.7: Accuracy of SVM for selected genes by MRMR.

The selected 100 genes from MRMR were input to different optimisation algorithms which used the cost function defined in Section 5.5. Each optimisation algorithm was ran 20 times. In each run, the algorithm was iterated 100 times. So in each run after 100 iteration, the trained classifier' model for the final selected genes was used to examine the performance of the SVM model on unseen data (on the set of 25%) and the performance was measured in terms of sensitivity, specificity, and accuracy. These values were recorded and after 20 run, the means and standard deviations for these values were computed. Tables 5.4, 5.5, and 5.6 give information on the performance of each optimisation algorithms after 20 run (each run 100 iteration) for prostate, leukaemia, and lymphoma cancer datasets respectively. Furthermore, in the following tables, the means and standard deviations of the selected genes by each algorithm are provided.

Table 5.4: Means and standard deviations for the number of selected genes, sensitivity, specificity, and accuracy of SVM classifier for 20 runs of optimisation algorithms for prostate cancer dataset.

	Number of Genes		Sensitivity		Specificity		Accuracy	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
GA	41.80	3.22	98.40	1.80	98.72	1.08	97.35	2.08
PSO	35.90	5.03	97.57	2.43	98.96	1.45	97.40	2.61
HS	28.20	4.53	98.73	1.99	98.72	1.36	98.09	2.63
COA	16.70	4.83	98.85	1.79	99.06	1.29	98.70	2.68
COA-HS	8.40	3.12	99.33	1.96	99.92	1.36	98.97	2.37

Table 5.5: Means and standard deviations for the number of selected genes, sensitivity, specificity, and accuracy of SVM classifier for 20 runs of optimisation algorithms for leukaemia cancer dataset.

	Number of Genes		Sensitivity		Specificity		Accuracy	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
GA	29.30	2.86	99.32	1.42	99.51	1.12	98.20	1.61
PSO	14.50	3.40	99.17	1.87	99.33	1.24	98.85	1.92
HS	31.10	3.73	99.64	1.57	99.54	1.19	98.41	1.67
COA	8.00	2.93	99.31	1.34	99.87	1.04	99.29	1.54
COA-HS	6.50	2.72	99.42	1.18	99.61	1.23	99.36	1.36

Table 5.6: Means and standard deviations for the number of selected genes, sensitivity, specificity, and accuracy of SVM classifier for 20 runs of optimisation algorithms for Lymphoma cancer dataset.

	Number of Genes		Sensitivity		Specificity		Accuracy	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
GA	28.80	2.81	99.40	0.87	99.51	0.78	98.70	1.12
PSO	13.40	2.98	99.49	0.93	99.64	0.88	98.92	1.19
HS	29.70	3.33	99.17	1.87	99.33	0.92	98.05	1.94
COA	7.70	2.16	99.36	1.74	99.61	1.16	99.27	1.84
COA-HS	5.10	1.99	99.91	0.90	99.87	0.71	99.45	1.04

Overall, COA-HS and COA outperformed GA, PSO, and HS in that these two algorithm selected significantly less number of genes while achieving better means for accuracy, sensitivity and specificity. In respect to prostate cancer dataset, as can be seen if Table 5.4 after 20 run of COA-HS a mean of 8.4 genes was selected to achieve a mean of 98.92 % for accuracy which is the highest accuracy acquired across different algorithms employed in this research. However, it is noted that in this dataset, GA had smaller standard deviations for accuracy and specificity when compared to other algorithms. Regarding leukaemia dataset, COA-HS outperformed other algorithms in most cases apart from the mean specificity and its standard deviation, where COA had a better performance (See Table 5.5). Finally, in Table 5.6 it can be seen that COA-HS achieved slightly better results when compared COA in all criteria and had significantly higher performance when compared to HS, GA, and PSO. It is noted that although GA had a mean of 28.8 for the selected genes, this algorithm had small standard deviations for all three classification performance measures when compared to PSO, GA and COA.

It was observed that in each run of an optimisation algorithm different combination of genes were selected and each algorithm tend to select different number of genes. In respect to the proposed algorithm, COA-HS, the final selected genes after each run were recorded and after 20 run the genes were ranked based of how many times they were repeated for each dataset. In the following the genes which were selected at least 10 times after 20 run of the COA-HS algorithm for each dataset are investigated.

In respect to prostate cancer, *37639_at* and *38087_s_at* were selected. *37639_at* was a probe-set for *Hepsin* gene, also known as *HPN*, is a gene that encodes a type II transmembrane serine protease. Expression of the encoded protein is associated with the growth and progression of prostate cancer [210]. Klezovitch et al., [211] demonstrated that hepsin was highly expressed by 10 fold in prostate cancer. *38087_s_at* is a probe-set for S100 calcium binding protein A4 gene. This gene is a protein coding gene and its gene ontology annotation associated with poly (A) RNA binding and identical protein binding. This gene has been selected as a signature for prostate cancer classification in many studies [212–214].

Regarding leukaemia dataset, *Zyxin* gene was found to be selected at least 10 times out of 20 runs of COA-HS algorithm. This gene is a focal-adhesion-associated phosphoprotein that involves in the control of actin assembly. Literature suggests that in the signal transduction pathway this gene could act as a messenger that control the adhesion-stimulated changes in gene expression [215]. Several studies have identified this gene as prominent in leukaemia cancer classification [26,216–218].

For lymphoma dataset one gene was found that at least was selected 10 times out of 20 runs namely *GENE1296X* gene. This gene is known as *MCL1* gene which is a protein coding gene that encodes an anti-apoptotic protein that is a member of the *Bcl-2* family. It is known that *Bcl-2* plays an important role in some cancers such as leukaemia and lymphoma [219]. This gene has previously been selected for its discriminatory power in lymphoma cancer classification [220].

5.7: Summary

In this chapter, a two-stage gene selection process using MRMR and the COA-HS algorithm was proposed in order to minimise the number of genes that could provide high accuracy in cancer classification. To this end, first MRMR was used to reduce the number of genes to 100, so that the computational time could be reduced for an optimisation algorithm. The 100 candidate genes were then used as an input for the second stage of gene selection, during which COA-HS was combined with the SVM classifier and acted as a wrapper gene selection method. The LOOCV method was used to evaluate the performance of our proposed method, and the results were compared to other optimisation algorithms such as PSO, GA, HS, and COA. To account for overfitting, 75 % of data was used for training and cross-validation and the remaining 25% was used to report the performance of the classifier model.

Each optimisation algorithm was ran 20 times and in each run the algorithm iterated 100 times. The means and standard deviations fore sensitivity, specificity, and accuracy of SVM classifier for each algorithm were computed. The results suggested that the COA-HS outperforms other optimisation algorithms in reaching a higher classification performance whilst selecting the least number of genes among other optimisation algorithms.

Chapter 6: Gene Expression Analysis using RNA-Seq Data

In this chapter, first an overview of RNA-Seq data analysis will be explored in Section 6.1. Then in Section 6.2 a state-of-the-art pipeline for RNA-Seq analysis will be investigated.

6.1: Overview of RNA-Seq Data Analysis

As it was discussed in Chapter 1, RNA-Seq overcomes several limitations of microarray technology when measuring gene expression and more recently, has therefore become a popular choice for measuring gene expression. There are several steps towards a successful RNA-Seq data analysis, including experimental considerations in design, pre-processing and quality assessment, alignment, building a count table, normalisation, and downstream analysis.

6.1.1: RNA-Seq Experimental Considerations

In order to accurately answer a biological question, adequate information should be provided within a RNA-Seq experiment. For this reason, several experimental considerations need to be addressed, such as sequencing depth and the number of replicates. Sequencing depth for a sample refers to the number of reads that have been sequenced for the sample. Research suggests that there is a direct relation between the number of transcripts that can be discovered, and the depth of sequencing [75]. However, the biological question is the main factor in defining the adequate number of reads for a valid analysis. For instance, around five million mapped reads would be sufficient to identify the highly expressed genes, compared

with a range of 100 million reads that might be adequate if low expressed genes needed to be quantified [239]. Furthermore, for some studies, a few thousand reads is adequate, such as the 20 thousand reads that were used for splenic tissue to successfully differentiate the cell types [240].

There are two types of replicates in RNA-Seq, technical and biological replicates. In general, the number of replicates depends on both biological and technical variability. Although some studies show that increasing the depth of sequencing can improve transcription identification [241] and quantification of gene expression [242], others suggest that by sequencing less reads and increasing the number of replicates in an experiment, a greater statistical power can be achieved [243].

In order to design RNA-Seq experiment, an optimum number of replicates and sequencing depth should be calculated, where tools like Scotty fulfil this objective [244]. Scotty calculates the variability between replicates and the frequency at which new RNAs are quantified by utilising prototype data. T-test is used to estimate the power and sample size. Furthermore, empirical distributions can also be taken from publicly available datasets that are pre-loaded in Scotty [244]. To model the power first empirical observations are used to select theoretical distributions. These distributions are then used to fit the observed data. The software estimates the variance between different replicates from same condition which essentially is the determinant factor on deciding the number of replicates required. Busby *et al.*, [244] argue that although there exists a substantial heterogeneity among different experiments such that biological variation is less than technical variations, the estimate for sample size is more accurate if users supply Scotty with their own data.

6.1.2: Pre-Processing of RNA-Seq Data

As discussed in Sections 2.3.4 once the RNA-Seq short reads are sequenced by an NGS platform, the output of such platform is usually in a FASTQ format. The first step upon receiving FASTQ files is pre-processing, which is vital for removing technical and biological contaminations from the data, so that one can investigate more interesting variations from the datasets. Technical contaminations include low quality reads and technical sequences, such as adaptors. Concerning the read quality, the PHRED score is used as a standard measurement, and ranges between 0-40. In general, read quality increases towards the 5' end of the reads, and bad quality reads are observed towards the 3' ends. For a valid analysis, reads with a Phred score of less than 20 should be removed, as these reads introduce errors and lead to noise in read counts. Three popular tools to aid in the visualisation of read quality and other important metrics for NGS data are the FASTQC [245], NGSQC [73], and HTQC

[246] software. In order to remove bad quality reads, one can use software such as FASTX [247], Cutadapt [248], or Trimmomatics [249] to trim out the bad quality reads based on a given threshold.

With regards to technical sequences like adaptors, it is also essential to remove these sequences before the mapping step, especially if the reads are mapped to a reference genome. This is due to the fact that adaptors contain sequences from similar nucleotides to that of the organism sequences that are introduced artificially, which therefore could hinder the ratio of mappability and consequently create artefacts in the downstream analysis. Mappability refers to the state of being mappable for the reads that can be mapped to a reference genome. It is important to note that in the case of RNA-Seq data, duplicate reads are often observed. This is normal, as they can be the results of highly expressed genes, and not due to a PCR amplification step. Therefore, it is safe to ignore the duplication level in the RNA-Seq quality control step. Biological contaminations include the presence of polyA tails, rRNA, and mtDNA. Since up to 95% of RNA is rRNA, it is essential to carefully remove rRNA and concentrate on the remaining 5% of mRNA, which can result in a meaningful downstream analysis. A popular tool to remove rRNA content is the SortMeRNA toolkit [250].

6.1.3: RNA-Seq Alignment

In RNA-Seq, in order to find the locations of short reads, the sequenced reads must be aligned to a reference genome or a transcriptome assembly [251]. Mapping RNA-Seq reads is particularly challenging, as in most cases the reads are formed from mRNA and not DNA, which means some reads might overlap an exon-exon junction, at which the location's intron has been removed [89]. If RNA-Seq reads are aligned to a reference genome, it provides more information to discover novel transcripts and isoforms. However, in this method, reads should be able to be split, as some reads might be mapped to two exons (see Figure 6.1). This method is done by spliced-aware aligner software, either by using prior information of exon/intron boundary annotations, which are usually available to download in a GTF format from the Ensembl website, or without this information which is known as *de novo* spliced alignment [252]. It is noted that by supplying the GTF file, the quality of alignments could be improved significantly.

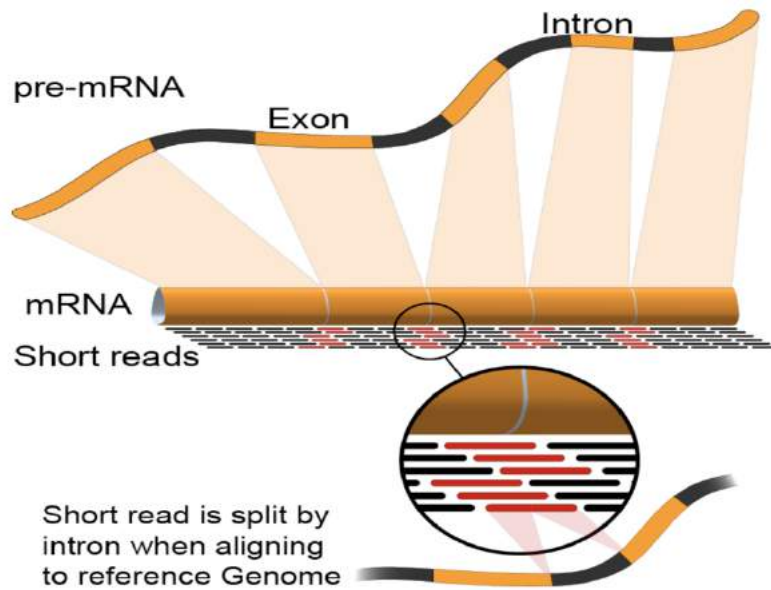


Figure 6.1: Junction reads [253].

If transcription discovery is not the objective of RNA-Seq, the sequenced reads can be aligned to a reference transcriptome, which is fast and useful for transcript quantification and is limited to identifying known exons and junctions. This method is done by unspliced aligner software, which aligns the reads to a reference transcriptome without allowing any large gaps. Finally, if a reference genome or transcriptome are not available, the alignment can be done by de novo assembly of the transcript sequences using de Bruijn graphs [254]. Figure 6.2 summarises different methods for RNA-Seq alignment.

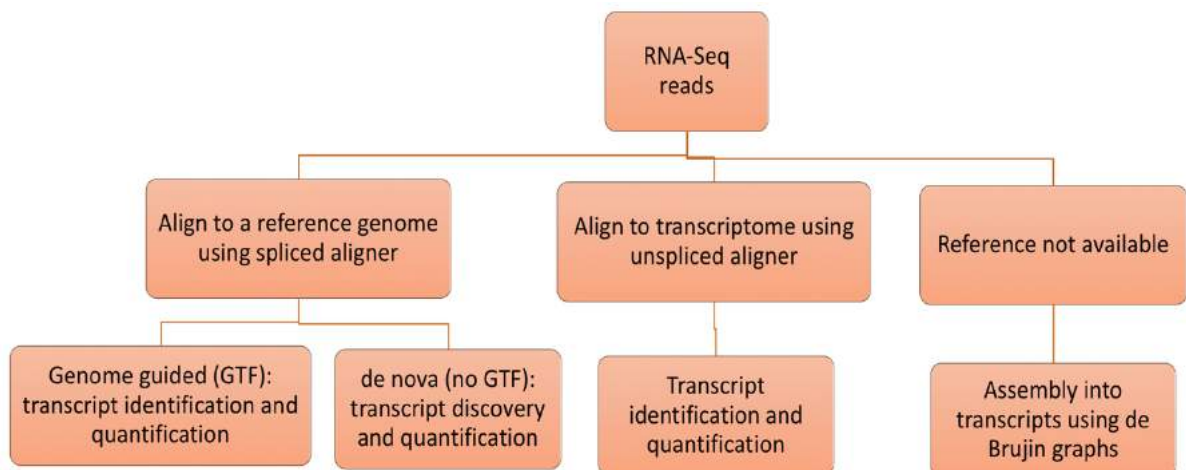


Figure 6.2: RNA-Seq alignment methods.

6.1.4: Creating a Count Table

After reads are successfully mapped to a reference, the read alignment information is usually presented in a SAM format, which stands for sequence alignment/map. However, this information is then converted into BAM format, which is the binary version of the SAM format, in order to reduce the file size and index its content better. Since a BAM file only contains the genomic locations of the reads, in order to count how many reads are mapped to unique regions, a list of genomic features (e.g. genes or exons) containing the start and end positions of such regions are required.

One simple method is to count the number of reads for every exon of each gene [21,75]. However, this method can ignore the reads that are mapped to other places than annotated exons [251]. Another method to quantify the reads is to count the reads along the total length of the gene, so that all reads from the coding sequences will be counted [255]. A number of R/Bioconductor packages, such as GenomicAlignments [256], Rsubread [257], and EasyRNASeq [258] can be utilised to obtain the count table. Furthermore, a popular python based software called HTSeq [259] can be used to achieve this objective. As a result, a matrix is formed in which each column corresponds to a sample, and each row corresponds to a genomic feature and its corresponding counts. The first column specifies a list of genomic features, and the rest of the columns specify the number of counts for genomic features (see Figure 6.3).

$$\text{Count table} = \begin{bmatrix} \text{Features} & \text{Sample 1} & \dots & \text{Sample } n \\ F_1 & X_{1,1} & \dots & X_{1,n} \\ F_2 & X_{2,1} & \dots & X_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ F_m & X_{m,1} & \dots & X_{m,n} \end{bmatrix}$$

Figure 6.3: Count table for RNA-Seq.

As the number of reads that overlap a gene is directly related to the length of transcripts, all of these methods can encounter the same problem due to initial random RNA fragmentation. Therefore, a normalisation step based on transcript length and sequencing depth is essential, which will be discussed in the next section.

6.1.5: Normalisation

In order to remove biases and artefacts that can affect the downstream analysis the normalisation step is an essential task to be carried out on the RNA-Seq count table. Gene

length bias is the first issue that should be accounted for in the normalisation task. For example, if the number of counts for gene n is 30 reads, and for gene m is 60, at first it can be inferred that gene m is expressed more than gene n . However, if genes n and m have lengths of 30 and 60 bp respectively, it should be noted that both genes actually have the same level of expression.

The next bias is the library size, which corresponds to the total number of reads for each sample. As illustrated in Table 6.1, if the total number of reads for replicate two is double those of replicate one, although each gene in replicate 2 might appear to have an expression twice those of gene 1, in fact none of these genes are differentially expressed.

Table 6.1: Library size affect.

	Replicate 1	Replicate 2
Gene 1	10	20
Gene 2	20	40
...
Gene n	30	60
Total Reads	1000	2000

A simple method to correct such a bias is to plot both replicates against each other and calculate the slope. Ideally the slope of such a plot should be 1, however if it deviates from 1, one can normalise such biases by using the obtained slope number. Other biases such as GC content and batch effect are also important to take into consideration.

In order to overcome these biases, several normalisation methods have been proposed, including reads per kilobase per one million mapped reads (RPKM) [75], DESeq [260], quantile (Q) [261], total count (TC), upper quantile (UQ) [262], trimmed mean of M-value (TMM) [263], and median normalisation methods. The most commonly used method for single-end reads is RPKM. In the case of paired-end reads, a similar approach called FPKM (fragments per kilobase per one million mapped fragments) is used, so that the two reads that come from one fragment are counted as one [264]. RPKM simply normalises the reads for each gene through the following expression:

$$RPKM = \frac{\text{Number of reads for the gene}}{\text{Gene length (in bp unit)} \times \text{Library size (in million reads unit)}} \times 10^9 \quad (6.1)$$

Dillies *et al.*, performed a comparative analysis between different normalisation methods [265] in terms of their power in intra-variance (group variance), count distribution, clustering, and false positive rate. In their study the performance of different methods was reported in terms of *not satisfactory*, *satisfactory*, and *very satisfactory* (see Table 6.2). The results suggest that the RPKM and TC methods are not very suitable if large differences exist in library size (count distribution). In contrast, DESeq and TMM methods provided very satisfactory results. These two methods are used in two popular R/Bioconductor packages called DESeq [260] and EdgeR [266] respectively.

Table 6.2: Summary of results for seven normalisation methods; 0 indicates not satisfactory, 1 indicates satisfactory, and 2 denotes very satisfactory (modified from [265]).

<i>Method</i>	<i>Distribution</i>	<i>Intra- Variance</i>	<i>Housekeeping</i>	<i>Clustering</i>	<i>False Positive Rate</i>
<i>TC</i>	0	1	1	0	0
<i>UQ</i>	2	2	1	2	0
<i>MED</i>	2	2	0	2	0
<i>DESeq</i>	2	2	2	2	2
<i>TMM</i>	2	2	2	2	2
<i>Q</i>	2	0	1	2	0
<i>RPKM</i>	0	1	1	0	0

Overall, if one wishes to find differentially expressed genes within the same sample, RPKM can provide satisfactory results and is more simple. However, if the objective is to find differentially expressed genes across different samples, the DESeq and TMM methods can provide very satisfactory results.

6.1.6: Modelling Raw Counts, Dispersion and Differential Gene Expression

From a biological view point, a very interesting question is 'which genes are differentially expressed across different conditions?' One way to investigate this question is to look at the number of counts for each genomic location, for example genes, isoforms, or transcripts. Research suggests that the number of counts for a gene is a good indication of the abundance of that gene. However, when comparing this between different conditions (healthy vs cancerous), the observed counts are done separately. In this respect, statistical tests should be performed to see if the differences in read counts between two conditions are actually significant, or observed due to natural random variation. It is noted that if reads are sampled

independently, the number of observed reads would follow a multinomial distribution that is known to be well approximated by Poisson distribution. Several studies have used this model to identify differentially expressed genes [267,268]

The Poisson model provides a useful tool for estimating the probability that a read from condition one could map to a given gene, as well as estimating the probability that a read in condition two could map to the same gene. Consequently, one can tell in which condition the probability of observing more reads for a given gene is higher, which leads to the concept of differential gene expression. In the Poisson distribution model, the variance is equal to the mean, and this makes this model very simple as there is no need to estimate the variance [267]. However, it is important to investigate whether this distribution is feasible for RNA-Seq datasets or not, due to the fact that several sources of noise exist in such datasets. One source of noise is referred to as shot noise, and denotes the existence of variance in counts [260]. It is known that this follows a Poisson distribution. Standard deviation (σ) of such noises is equal to the square root of the mean count (μ). Another source of noise is the sample noise that includes biological and technical noises. Research suggests that the Poisson model performs well for technical replicates, as the variance is equal to the mean [21]. However, this research suggests that in the case of biological replicates, actual variance could be predicted inaccurately by the Poisson distribution model, since genes with higher mean counts have a higher variance than the mean, which can lead to an increase in false discovery rates (type1 errors). This phenomenon is referred to as an overdispersion problem in the literature [269,270].

Negative binomial distribution has more recently been used in order to overcome the overdispersion problems that the previous model encountered [260,266]. This model includes an extra parameter that accounts for dispersion, and can be seen below.

$$y = \text{NegativeBinomial}(ab, 1/a) \quad (6.2)$$

where ab is the mean (μ), and $(1/a)$ is the dispersion parameter (\emptyset). In this distribution, variance can be calculated by the following expression:

$$\text{variance}(y) = v + v^2\emptyset \quad (6.3)$$

In the above expression, v accounts for shot noise, which is related to Poisson sampling noise, and $v^2\emptyset$ accounts for technical and biological noise. Usually the first step towards differential expression analysis when dealing with biological replicates is to estimate these

parameters. In order to have a good estimation for these parameters, all methods that are proposed for identifying differentially expressed genes make some assumptions about the form of underlying distribution. This is due to the fact that the number of samples are small, and without assumptions, the correct estimation would be impossible [271].

Two of the well-known R/Bioconductor packages that use the negative binomial approach are edgeR [266] and DESeq [260]. However, these methods differ by which the dispersion parameters are estimated, as well as using different hypothesis testing approaches to find differentially expressed genes.

In DESeq, the number of reads for gene i in sample j (R_{ij}) is modelled via negative binomial distribution as shown by following expression:

$$R_{ij} = \text{NegativeBinomial}(\mu_{ij}, \sigma_{ij}^2) \quad (6.4)$$

where μ_{ij} is the mean and σ_{ij}^2 is the variance. Initially, μ_{ij} is calculated, which is proportional to a size factor (S_j) that accounts for the sequencing depth, multiplied by a variable that accounts for the gene expression number for gene i in sample j (Q_{ij}). Next, the variance is calculated by the following expression:

$$\sigma_{ij}^2 = \mu_{ij} + S_j^2 v_{i,p(j)} \quad (6.5)$$

where $v_{i,p(j)}$ is the per gene raw variance and is the smooth function of Q_{ij} . As the general form of a negative binomial, the above expression also accounts for shot noise and biological variations.

In the edgeR Bioconductor package [266], data is also modelled via negative binomial distribution, as shown by the following expression:

$$Y_{gi} = \text{NegativeBinomial}(M_i P_{gj}, \phi_g) \quad (6.6)$$

where M_i is the read counts, P_{gj} is the relative abundance of gene g in condition j to which sample i belongs, and ϕ_g is the dispersion. $M_i P_{gj}$ is equal to the mean (μ_{gi}), and the variance can be calculated as follows:

$$\text{variance}(Y_{gi}) = \mu_{gi} + \mu_{gi} \phi_g \quad (6.7)$$

Once the parameters of mean and variance are modelled, a generalised linear model (GLM) is fitted into the data to get the variance-mean dependence. Figure 6.4 illustrates the fitted line for mean and variance by the Poisson (purple line) model, DESeq (orange line), and edgeR (dashed orange). Generally, in all fitted lines in Figure 6.4 as the mean count increases, the variance also increases. However, the fitted line from Poisson model is linear which fits well only for the lower mean counts and poorly estimates mean and variance for the higher mean counts. In contrast, the fitted lines by edgeR and DESeq are nonlinear and fit well for all ranges of mean and variance [260].

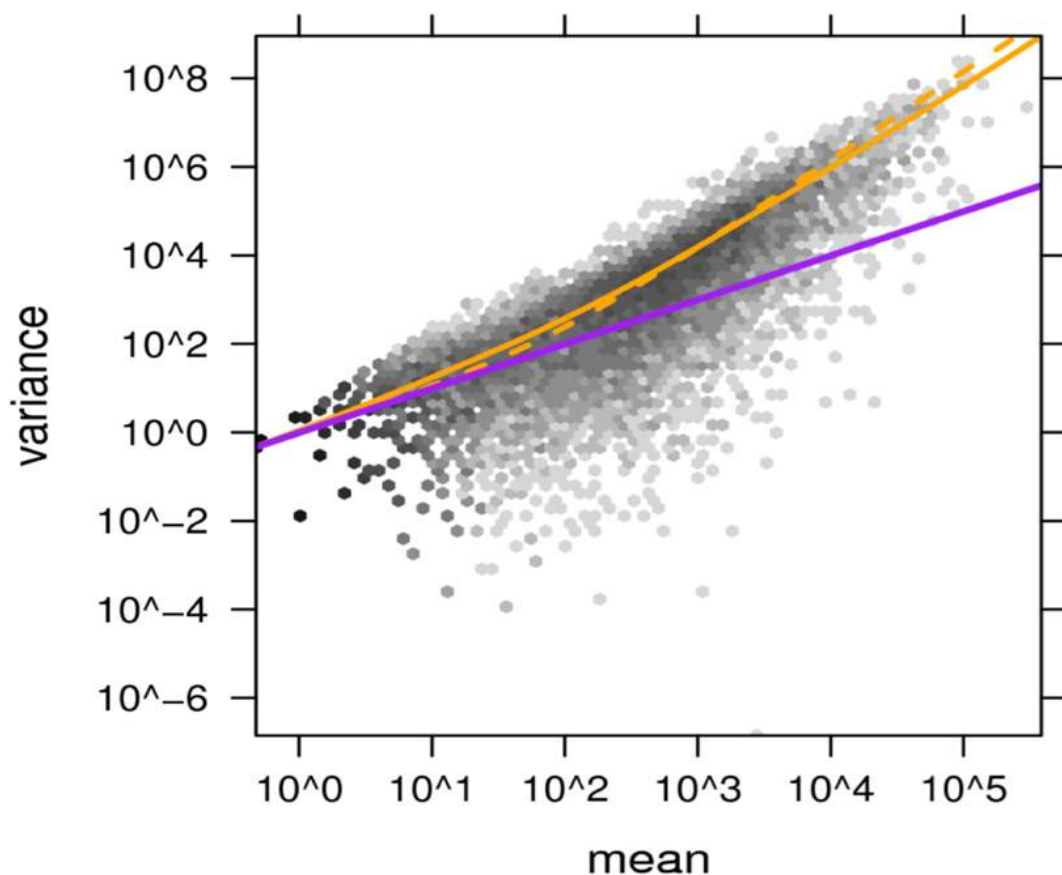


Figure 6.4: Variance-mean dependence adapted from [260].

In order to identify differentially expressed genes, the null hypothesis of $\mu_1 = \mu_2$ is investigated, where μ_1 and μ_2 are the mean expression values for conditions 1 and 2 respectively. For both conditions 1 and 2, the summation of reads across all replicates (k_{i1} and k_{i2}) for each gene is calculated. The overall sum is equal to $k_{i1} + k_{i2}$. Finally, the probability of observing the actual sum is calculated using a Wald test. The Wald test provides p-values for each gene, and based on a threshold, one can determine differentially expressed genes. Most packages now implement the Benjamini-Hochberg procedure to control the FDR.

6.1.7: Alternative Splicing Analysis

So far, the statistical methods for differential gene expression were investigated. However, RNA-Seq data provides information that sheds light on differential analysis at the transcript level and alternative splicing. Each gene contains several transcripts, and each transcript from a gene can differ from other transcripts from the same gene in its starting and ending sites, as well as differing from the inclusion of exons. The translation of different transcripts results in different protein structures and functions, which leads to the importance of the transcript's expression for the phenotype of cells and investigating diseases. The advances in informatics approaches have paved the way to look into differential exon usage and differential isoform expression.

A pioneering approach to identify differential exon usage is the DEXSeq [272] method, which is implemented in a R/Bioconductor package. Simply put, the differential exon usage approach identifies the exons that are expressed differently across different conditions within each gene. In this method, the relative usage of each exon is used in order to identify the conditionally specific usage of exons, which can be calculated by the following expression:

$$Usage\ of\ exons = \frac{Number\ of\ reads\ mapping\ to\ the\ exon}{number\ of\ reads\ mapping\ to\ other\ exon\ for\ same\ gene} \quad (6.8)$$

DEXSeq has a similar approach to the DESeq package in finding differential exon usage, where the main steps are count normalisation, dispersion estimation, and differential testing that returns a p-value. However, one of the major differences is preparing annotations that can allow counting reads that overlap exons. For this reason, after the reads are mapped to a genome reference, an annotation file with exon coordinates is used to count the number of reads in each exonic location. Since some exons can be seen more than once in an annotation file due to their inclusion in multiple transcripts, they can overlap each other. In such cases, in order to make sure each exon is counted only once, DEXSeq uses a flattened form of annotation file in which exons from the same coordinates are flattened into counting bins [272] (See Figure 6.5).

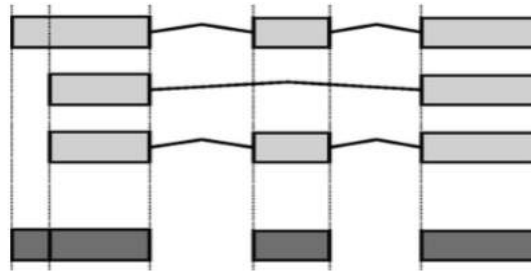


Figure 6.5: Flattened exons' locations.

In contrast to differential exon usage, where the focus is at the exon level, in differential isoform expression, the unit of study is the isoform and the objective is to identify isoforms that are expressed differently across conditions for the same gene. Zheng and Chen [273] proposed a method based on the hierarchical Bayesian model called BASIS to provide a platform for differential expression at the isoform level across two conditions. A more popular tool that allows differential analysis at the transcript level for more than two samples is Cuffdiff [274]. Similar to DESeq and DEXSeq, Cuffdiff also hypothesises that the number of reads that are mapped to a transcript is proportional to its abundance, and uses a negative binomial model for read counts. Furthermore, in the Cuffdiff method, a scaling factor is used to correct the sequencing depths across different samples, and the Benjamini-Hochberg method is used to control for FDR. This method produces very accurate results, as both biological and technical variations are taken into consideration in its statistical model to look for differential isoform expression [274,275]. More recently, a new approach has been proposed that allows the identification of differential gene expression and differential isoform expression at the same time using a hierarchical likelihood ratio test [276].

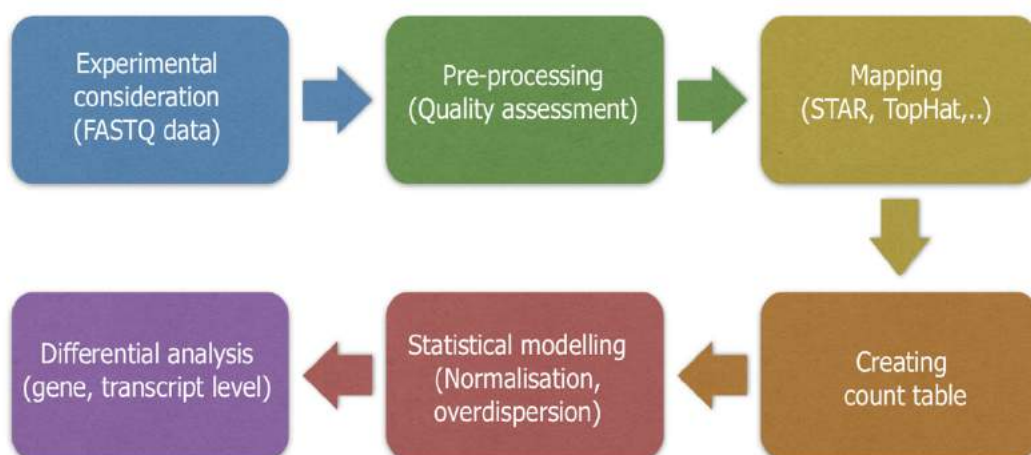


Figure 6.6: RNA-Seq data analysis.

6.2: Pipeline for Analysis of RNA-Seq: a Case Study

In this Section, a state-of-the-art pipeline for RNA-Seq analysis will be investigated. Figure 6.7 illustrates the steps in this pipeline. Whilst there have been numerous studies to improve RNA-Seq data manipulation, quality control, and downstream analysis, the analysis of RNA-Seq when examining gene expression remains challenging compared to microarray data. Since the methods to analyse NGS data in general are essentially statistical approaches, R/Bioconductor, a free source software, has become a popular tool to implement such analysis. There are several reasons for the popularity of performing NGS analysis in R/Bioconductor. For example, this software is easily accessible due to its affordability. Furthermore, it is a platform assisting with the statistical challenges of NGS data, and can be used for annotation and handling large datasets.

There are numerous Bioconductor packages to use for RNA-Seq analysis. This has led to the suggestion of several pipelines for this kind of analysis. However, due to the rapidly developing nature of statistical approaches for RNA-Seq, the proposed pipelines have also undergone several changes to improve their results. In this chapter, we investigate a state-of-the-art pipeline for pre-processing and analysis of RNA-Seq that can pave the way to extract biologically relevant results from large datasets. Figure 6.7 illustrates the steps required for this pipeline. Usually RNA-Seq data is in the format of FASTQ, and the first step towards a successful downstream analysis is the pre-processing step. The pre-processing step is divided into four processes that are shown in orange in Figure 6.7. Once the qualities of reads are satisfactory, the alignment step follows. Finally, different downstream analysis, such as differential gene expression, differential exon usage, GO and pathway analysis, and classification approach are performed.

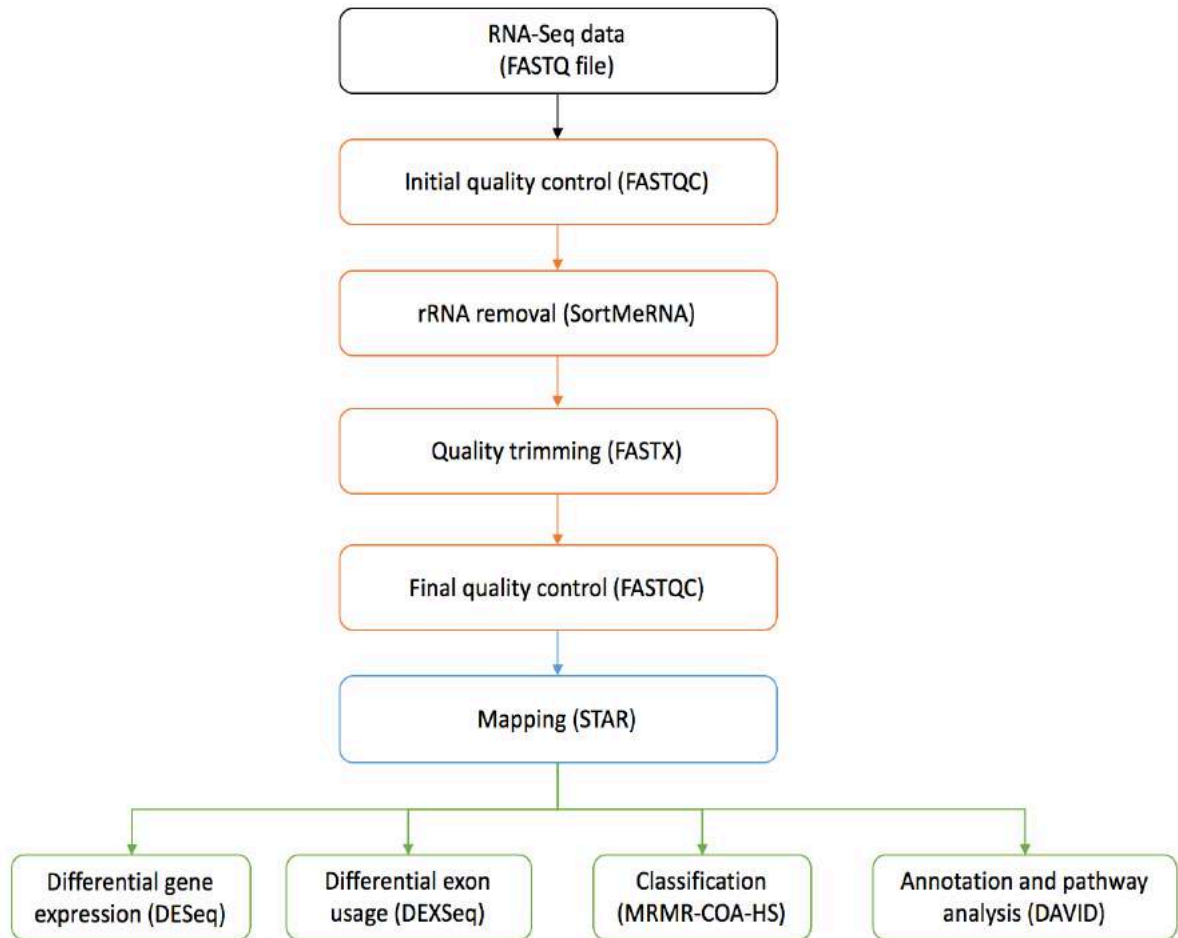


Figure 6.7: RNA-Seq analysis workflow.

6.2.1: Utilised RNA-Seq Data

It is observed that a mutation in the aryl hydrocarbon receptor interacting protein (AIP) gene occurs in familial isolated pituitary adenoma (FIPA). This leads to early onset acromegaly in patients, and in most cases, invasive pituitary adenoma forms as a result. It is established that patients with positive for AIP have bigger body sizes than normal. Ascertaining pathogenicity of missense mutations is an abstruse task and to date around 70 AIP variants have been identified. However, it is still unclear how pituitary tumorigenesis can be caused by AIP inactivation. In this study, we used RNA-Seq data produced by Aflorei [277] at the Genome Centre of Queen Mary University of London.

Drosophila was investigated as a subject of interest by Aflorei [277]. The *Drosophila* AIP orthologue (CG1847) gene encodes a protein that resembles the human AIP. In brief, CG1847 defective flies were generated through in vivo RNAi knockdown to get a putative null allele of CG1847. In order to investigate the differentially expressed genes and underlying molecular

mechanisms as a result of the loss of AIP, RNA-Seq data was produced from mutant (3 samples) versus control (3 samples) male larvae.

To produce this RNA-Seq data, Aflorei first isolated total RNA using Qiagen RNeasy MicroKits and the samples were purified by the DNase I. Nanodrop was used to measure the RNA samples and Agilent 2100 bioanalyzer was used to examine the quality of the extracted RNA. Then, the RNA samples were normalized to 500 ng/μl and the normalised samples from both mutant and control were used to produce the cDNA libraries for the Illumina HiSeq 1500 platform. In this platform one lane was used to sequence all libraries and as a result around 30 million reads for each sample was acquired [277]. Each sample contained two files which corresponded to forward and reverse strands of short RNA-Seq reads.

6.2.2: Pre-processing

All analyses shown below are performed in Mac OS X with 16 GB of ram and an 8 core processor. The first step after receiving the RNA-Seq data (usually in FASTQ format) is to perform quality control assessments like examining the overall sequence quality, overrepresented reads, and the GC content of the data. A popular software to illustrate this information is FASTQC. To run FASTQC, all FASTQ files should be saved in one folder, and then the directory of the terminal is changed to that folder. Afterwards, FASTQC can be run for FASTQ files by using terminal. For the purpose of demonstration, the pre-processing steps are only shown for one sample here, and the same procedure can be repeated for all samples.

```
fastqc -t 8 ForwardRead.fastq ReverseRead.fastq
```

-t 8 option takes advantage of multicore processor capability to speed up the time for analysis. Two files (a zip and an HTML file) are created for each FASTQ file as a result of this command. By opening the HTML files, different analysis metrics such as basic statistics, per base sequencing quality, per base sequence content, per base GC content, per sequence GC content, sequence duplication level, overrepresented sequences, and Kmer content can be observed. Details for each metric are well-explained on the developer's website [245]. Four of the important metrics for one of the mutated samples acquired from FASTQC are shown below.

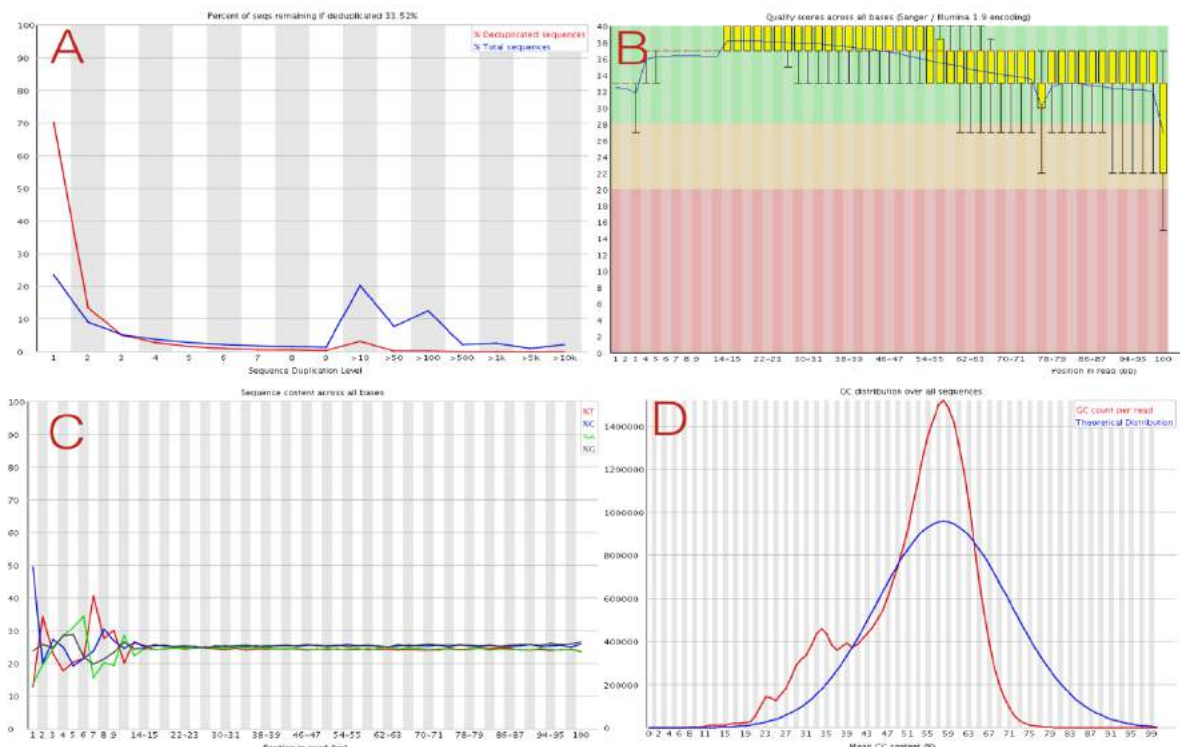


Figure 6.8: Initial FASTQC output.

First, one should consider quality trimming and adaptor removal if required based on FASTQC results. From the per base sequencing quality graph (Figure 6.8 B), it can be discerned that all reads have a quality higher than 20 based on the Phred score, therefore there is no need for removing any reads. However, if bad quality reads are observed, one can use software such as FASTX [247], Cutadapt [248], or Trimmomatic [249] to trim out the bad quality reads based on a given threshold. Since in the data that we received, adaptors had already been removed by the Genome Centre, the FASTQC metric on adaptor contamination indicates no contamination. In the presence of adaptors, the graph related to the adaptor contamination would identify the sequence of adaptor. By using software like FASTX and supplying the relevant sequence of the adaptor, one could remove them.

From Figure 6.8 C, it can be seen that there is a presence of noise in the first 10-12 nucleotides, which is in fact a universal bias from the Illumina RNA-Seq data. Since we are interested in differential gene expression and these biases are universal, these biases would cancel each other, and therefore one can safely ignore them. However, if the objective was to quantify gene expression, it would be essential to trim these nucleotides.

Figure 6.8 D shows the distribution of GC content per read. In this figure, the blue line presents the theoretical distribution of GC content by FASTQC, and the red line indicates the actual distribution of GC content. Ideally, the actual distribution should follow the theoretical distribution. However, as shown in this figure, the actual distribution presents a shoulder on

the left side of the graph. Practical observations suggest that if a shoulder is presented on the left side of the actual distribution, it is more likely to be due to the enrichment of A/T content. However, a shoulder on the right side is more likely caused by the presence of rRNA. Nevertheless, due to the nature of RNA-Seq data generation, rRNA will often be present. Therefore, in the next step, a software called SortMeRna [250] is used to account for rRNA molecules. SortMeRna performs the removal of rRNA in three steps. First, the forward and reverse reads of a given sample should be merged together as shown below:

```
merge-paired-reads.sh ForwardRead.fastq ReverseRead.fastq MergedReads.fastq
```

Then the resulting *MergedReads.fastq* can be used in the main command of SortMeRna as follows:

```
sortmerna --ref $SORTMERNA_DB --reads MergedReads.fastq --aligned MergedReadsWithrRNA --other MergedReadsWithoutrRNA --paired_in --fastx
```

where *SORTMERNA_DB* is the environmental variable for the SortMeRna database, and should be saved prior to the command. Argument "*--reads*" indicates the merged reads, "*--aligned*" indicates the reads which contain rRNA, "*--other*" represents those reads which are rRNA free, the "*--paired_in*" argument makes both paired reads goes to a single file, and "*--fastx*" indicates that the output file should be in the FASTQ format. In the next step, the merged reads (*MergedReadsWithoutrRNA*) that are rRNA free should be unmerged using the following command.

```
unmerge-paired-reads.sh MergedReadsWithoutrRNA.fastq ForwardRead.fastq ReverseRead.fastq
```

After this step, another assessment on the read quality should be made by the FASTQC software, in order to make sure satisfactory results are acquired through the following command:

```
fastqc -t 8 ForwardRead.fastq ReverseRead.fastq
```

It is noted that by using SortMeRna, the shoulder on the left side of figure 7.2 D has been reduced to less than 400,000 from 460,000 before using SortMeRna (see Figure 6.9). However, other metrics remained the same as expected. In this step, the initial total reads before applying SortMeRna was 31,497,483; and after SortMeRna the total number of reads was 30,797,917; resulting in a reduction of 699,566 reads.

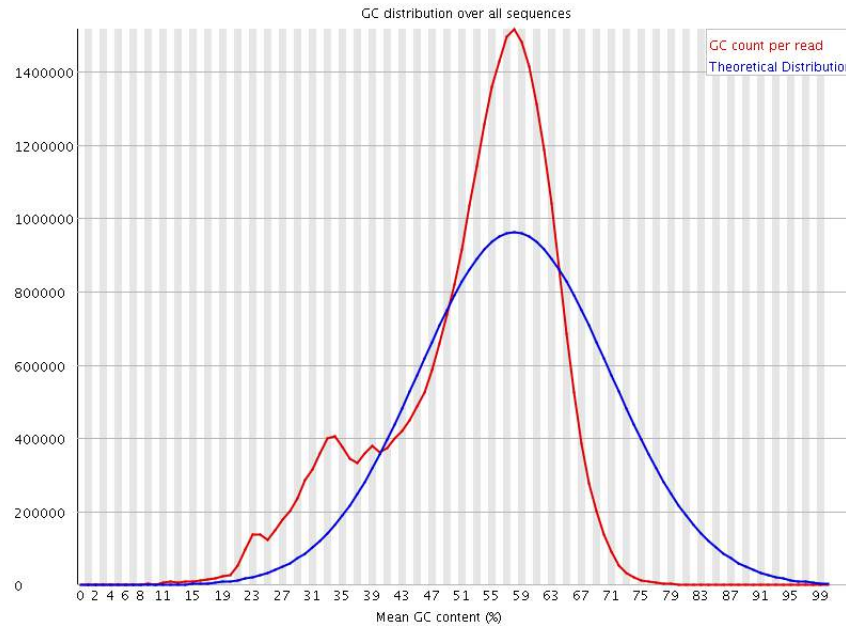


Figure 6.9: Per GC content for mutated sample after SortMeRna

Although the rRNA content has been removed, there is still a shoulder on the left side of the GC for the distribution. In this situation, it is advisable to check the overrepresented sequences in “Blast”, which is an NCBI utility [278] in order to reveal the nature of these sequences. By blasting these sequences, it becomes clear that these overrepresented sequences are mostly related to mitochondrial sequences. Since we will be mapping the reads to the *Drosophila* genome, the reads originated from mitochondria will not be mapped. Therefore, this step should be satisfactory for the pre-processing step.

6.2.3: Alignment of the Reads to a Reference Genome

After the pre-processing step is completed, and the quality of reads has been assessed to be satisfactory, the reads can be aligned to either a reference genome or transcripts. In this demonstration, the goal is to map the reads to a reference genome. Although several aligner software have been developed so far, there are two approaches that can be used to do the alignment task, which are the Burrows-Wheeler transformation (BWT) [279] and maximum exact matches (MEM) [280]. Since the objective is to align RNA-Seq to a reference genome, it is essential to use a spliced-aware aligner like TopHat [255] or STAR [281].

In this research, STAR, which is an ultrafast universal RNA-Seq aligner based on MEM protocol, is chosen to aid in the alignment process. First, a genome index should be generated so that the STAR software is aware of exon-exon junctions when doing the alignment. In order to generate the indexed genome, a reference genome in FASTA format, and a genome annotation file in a GTF format are required. The corresponding files for *Drosophila* are

downloaded from the Ensemble website [282]. The command to generate the indexed genome is shown below.

```
STAR |  
--runMode genomeGenerate |  
--genomeDir StarGenome |  
--genomeFastaFiles Drosophila_melanogaster.BDGP5.dna.toplevel.fa |  
--sjdbGTFfile Drosophila_melanogaster.BDGP5.dna.toplevel.gtf |  
--sjdbOverhang 100
```

The first parameter (`--runMode`) specifies the mode in which STAR should be run. The second parameter (`--genomeDir`) defines the output directory. The third and fourth parameters point STAR to the FASTA file and GTF file respectively. The last parameter (`sjdbOverhang`) denotes the sequencing read length, and a default value of 100 is used. Once this step is done, the genome index is saved in the directory specified by the second parameter (`--genomeDir`). This folder is used in the next step to align the reads to it as shown below.

```
STAR |  
--runMode alignReads |  
--genomeDir StarGenome |  
--readFilesIn ForwardRead.fastq ReverseRead.fastq |  
--sjdbGTFfile Drosophila_melanogaster.BDGP5.dna.toplevel.gtf |  
--outSAMtype BAM
```

where the third parameter (`--readFilesIn`) directs STAR to do forward and reverse reads of the sample, and the last parameter (`--outSAMtype`) specifies the output should be in the format of BAM. Once this command is successfully finished, several files will be created, including a file whose name ends with `Aligned.out.bam` that contains the alignment in BAM format. It is essential to sort this file by sequencing position and then index it for further analysis by using a tool called `samtools` [283] as shown below.

```
samtools sort -n Aligned.out.bam
```

```
samtools index Aligned.out.bam
```

This step concludes the alignment phase. The steps required for differential analysis at the gene, exon, and isoform levels differ from the point of counting the number of reads for each genomic location. This is due to the fact that different models are required to count the reads for different downstream analysis. Therefore, the analysis of differential expressions is divided into the three separate sections described in the next section.

6.2.4: Differential Gene Expression

Differential gene expression is divided into four subsections, which are counting reads over genes, normalisation, dispersion estimation, and testing for differential gene expression. The R/Bioconductor code written for this analysis can be found in appendix 1.

6.2.4.1: Counting Reads Over Genes

In order to count the number of reads over each gene, in addition to the aligned reads, a gene model is required and can be accessed in the GTF or GFF3 format on Ensembl website [282]. The latest release of the GTF file for Drosophila was acquired from the Ensembl website for this analysis (Drosophila_melanogaster.BDGP5.76.gtf). A transcript database (TxDb format) called TxDbFromGFF was then created from this GTF file using the GenomicFeatures package (makeTxDbFromGFF command). This database can be utilised to create a separate file for the genomic locations of interest that is ranged-based and includes genes, exons, and transcripts. Since we are dealing with exons in RNA-Seq, a list of exons for each gene (GRangesList format) is then created using exons with a command from the GenomicFeatures package, and saved as an ExonByGenes object. Each gene in the GRangesList is stored in the GRanges format. The length of the observed GRangesList for Drosophila is 15682; and a snapshot of the list acquired via R/Bioconductor is shown below.

```
> ExonByGenes
GRangesList object of length 15682:
$FBgn00000003
GRanges object with 1 range and 2 metadata columns:
      seqnames      ranges strand | exon_id exon_name
      <Rle>        <IRanges> <Rle> | <integer> <character>
[1]      3R [2648220, 2648518]   + |    47075   CR32864:1

$FBgn00000008
GRanges object with 13 ranges and 2 metadata columns:
      seqnames      ranges strand | exon_id exon_name
[1]      2R [18024494, 18024531]   + |    21079   FBgn00000008:1
[2]      2R [18024496, 18024713]   + |    21080   FBgn00000008:2
[3]      2R [18024938, 18025756]   + |    21081   FBgn00000008:3
[4]      2R [18025505, 18025756]   + |    21082   FBgn00000008:4
[5]      2R [18039159, 18039200]   + |    21087   FBgn00000008:5
...
[9]      2R [18058283, 18059490]   + |    21091   FBgn00000008:9
[10]     2R [18059587, 18059757]   + |    21092   FBgn00000008:10
[11]     2R [18059821, 18059938]   + |    21093   FBgn00000008:11
[12]     2R [18060002, 18060339]   + |    21094   FBgn00000008:12
[13]     2R [18060002, 18060346]   + |    21095   FBgn00000008:13

...
<15680 more elements>
-----
seqinfo: 15 sequences (1 circular) from an unspecified genome; no seqlengths
```

Figure 6.10: Exons grouped by gene in GRangesList format.

Once the previous step is done, a count table was created by using the `summarizeOverlaps` function from the `GenomicAlignments` package and saved as a `RangedSummarizedExperiment` object. This function can count the reads for each gene for all samples simultaneously, as this function accepts a file path to all of the BAM files that need to be processed. As a result, this function creates a file (`RangedSummarizedExperiment` format) containing three main components (see Figure 6.11), including samples' information, actual count matrix, and genomic ranges, which can be accessed using the `colData`, `assay`, and `rowRanges` commands respectively. The resulting count matrix has a dimension of 6 (samples) by 15682 (genes) for *Drosophila* datasets. Samples' information is primarily an empty component, and can be supplied using samples' information, such as samples' names and corresponding conditions in a character vector format.

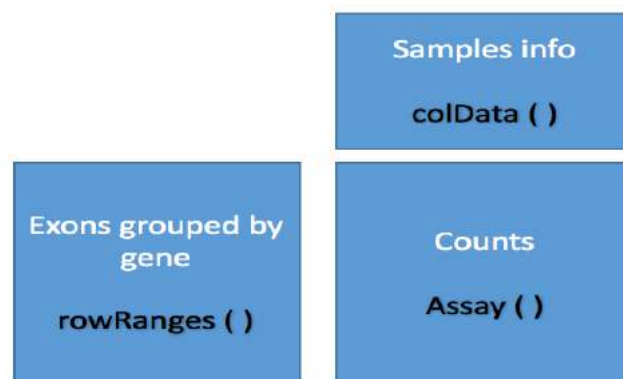


Figure 6.11: `RangedSummarizedExperiment` format.

Analysis of the literature suggests that the `DESeq2` package is mostly used for differential gene analysis, and therefore this package is chosen as the preferred method for this analysis [284]. In order to use `DESeq2`, the `RangedSummarizedExperiment` format should be converted into a `DESeqDataSet` class object that provides extra manoeuvrability in datasets, as the `DESeqDataSet` class has an argument called *design formula* that accounts for the group condition of a sample for further analysis.

6.2.4.2: Sample Normalisation and Visualisation

In order to make a valid conclusion about data through visualisation methods like PCA plots, the variations due to gene length and sequencing depth should be taken into account, which is done during the normalisation step. Based on the comparative analysis from Section 6.6, `DESeq` normalisation provides very satisfactory results, and therefore will be used for this purpose. As discussed in Section 6.1.5, a size factor that is directly related to the ratio of library sizes is utilised in the `DESeq` package to normalise the data. This means if all samples

are sequenced at the same level, a size factor equal to one will be allocated to all of the samples.

In order to calculate the size factors for all samples, the `estimateSizeFactors` function from the DESeq2 package was used on the DESeqDataSet object. Table 6.3 shows the information on the estimated size factor for each sample.

Table 6.3: Estimated size factor for each sample using DESeq.

	Control5	Control7	Control8	Mutated5	Mutated6	Mutated7
Size factor	1.09	1.06	0.97	1.00	0.87	1.03

In order to check whether the normalisation method is satisfactory, the density of mean counts for each sample was plotted, and if it is observed that the densities from all samples almost overlap each other, that is a good indicator for a successful normalisation outcome (see Figure 6.12).

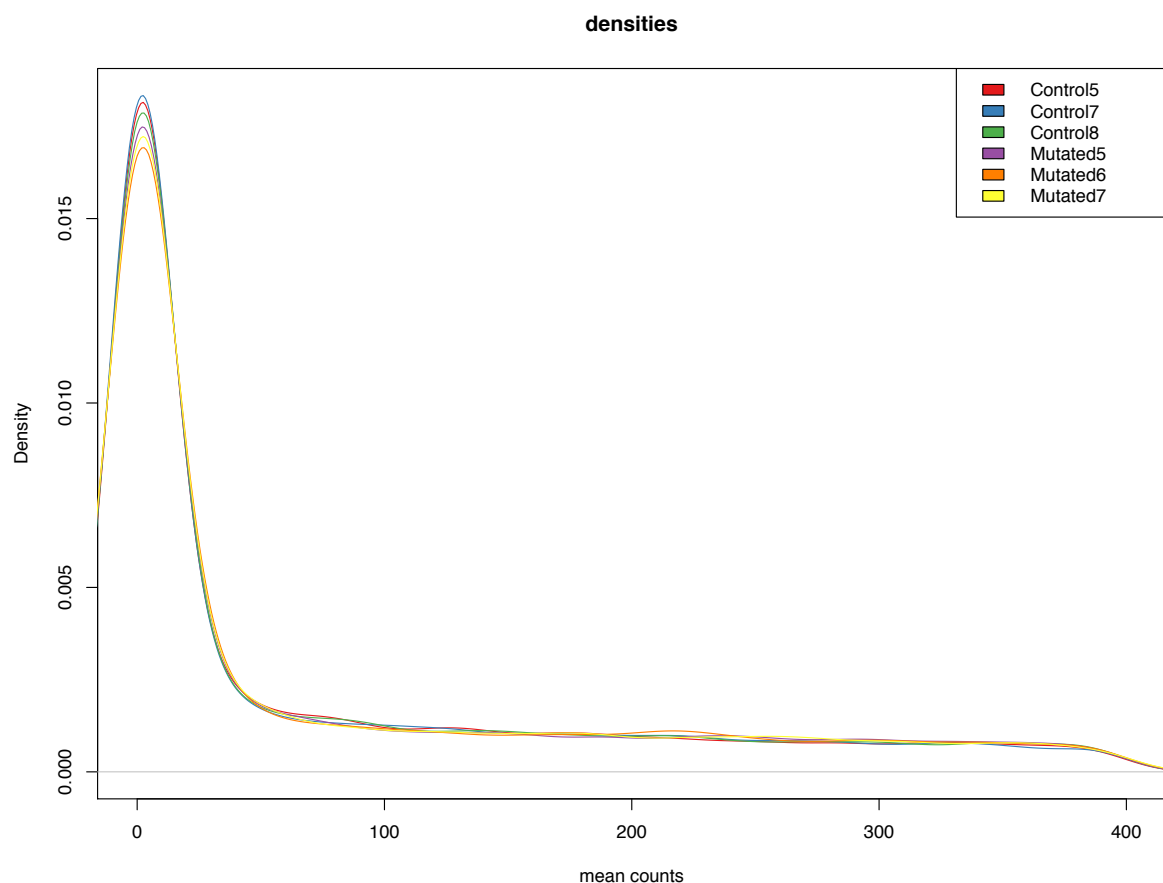


Figure 6.12: Density of mean counts for each sample.

Another way to validate a proper normalisation is to investigate the probability of observing a given number of counts from the datasets, which can be identified using an empirical cumulative distribution function (ECDF). Similar to density graphs, in ECDF graph samples should almost overlap each other in order to conclude that samples are normalised to provide satisfactory results. Figure 6.13 shows the observed probabilities for different mean counts from the ECDF function. It can be seen from Figure 6.13 that the observed probability for a given number of counts is almost similar for all samples, which provides evidence for a successful normalisation.

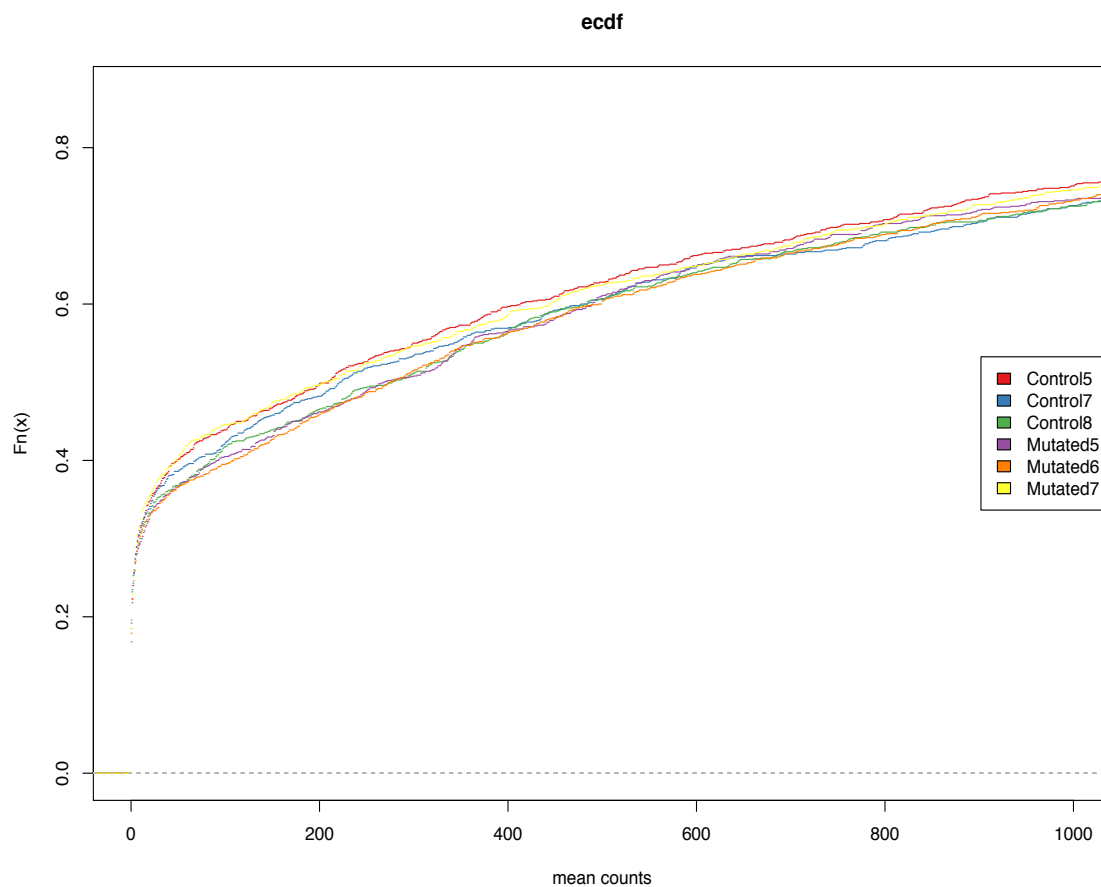


Figure 6.13: Probability of observing a given number of counts for all samples.

In order to see the similarities and differences between samples at this stage, a principal component analysis (PCA) can be very informative. However, PCA and other statistical approaches such as clustering are applied to homoscedastic data, which refers to a group of datasets in which the variance is the same for different ranges of mean values within a sample [285]. Due to the nature of RNA-Seq data, the variance increases with the mean value. In order to illustrate the difference in variance for different numbers of counts, two samples (control7 and control8) were plotted against each other (see Figure 6.14). In Figure 6.14,

each dot refers to a gene and it can be seen that genes with a higher number of counts have higher variances.

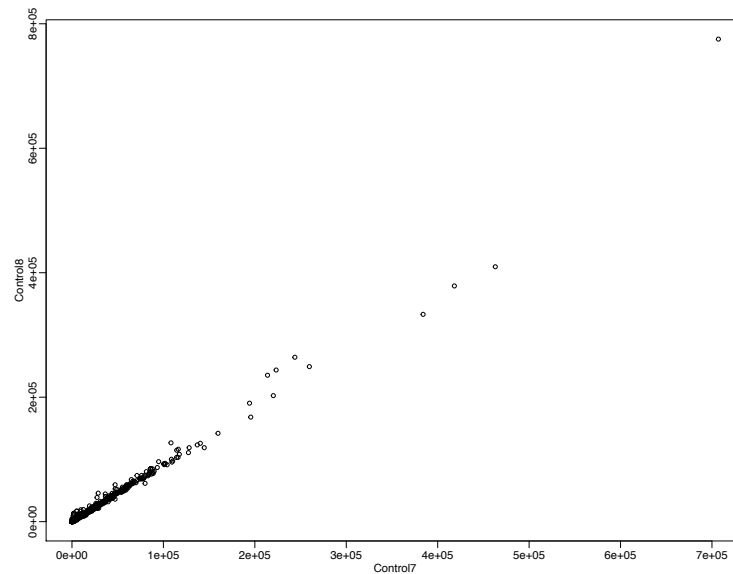


Figure 6.14: Natural scale for sample-sample visualisation.

One method to overcome this issue is to use a logarithmic scale instead of the normalised counts. As illustrated in Figure 6.15, this method provides a more constant variance across different ranges. However, by using this method, higher variation can be observed in the lower counts region. The variation in lower counts can be the result of Poisson noise for genes with lower counts, it can also be accounted for by the fact that the differences for genes with lower counts are maximised when a logarithm is applied.

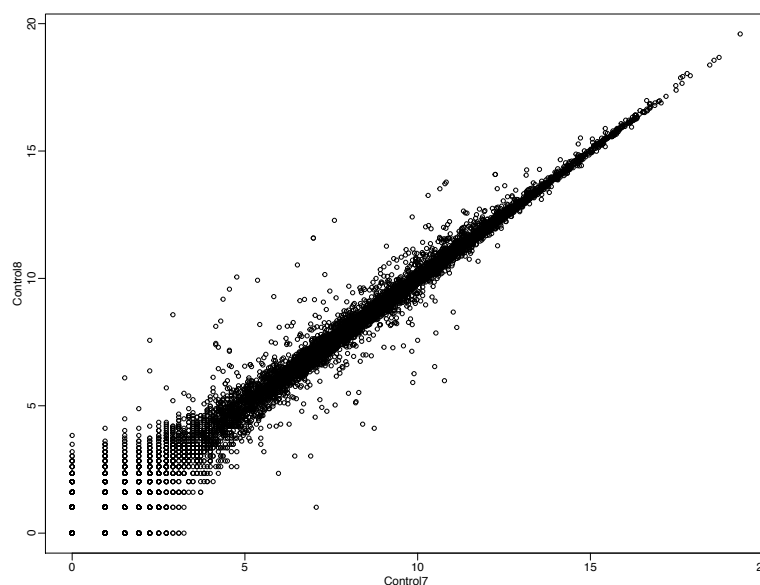


Figure 6.15: Log2 normalised counts scale for sample-sample visualisation.

The regularised logarithm transformation (rlog) and variance stabilising normalisation (vst) methods are useful approaches to correct for the variation in the lower counts region [284]. For genes with higher counts, rlog and vst act like a normal logarithm, however for lower counts of genes the values shrink. Figure 6.16 shows the effect of an rlog transformation on the counts. It can be seen that the rlog method stabilises the variance at different levels of the mean, and provides a satisfactory data format that prevents biases from affecting further analysis.

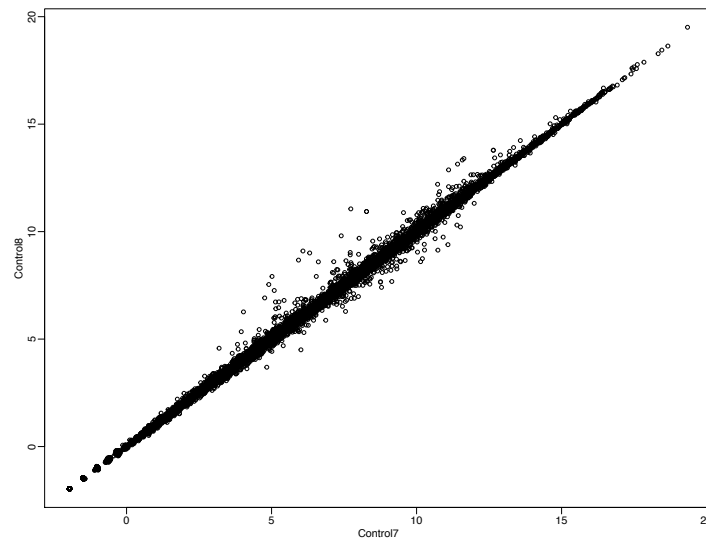


Figure 6.16: rlog scale for sample-sample visualisation.

Now that the datasets adhere to the characteristics for homoscedastic data, PCA can be applied to visualise the samples relations to each other. It can be seen from Figure 6.17 that a clear separation of two group conditions (control and mutated) is presented by using a PCA plot, which validates the experimental design for two conditions. In this plot, each dot represents a sample.

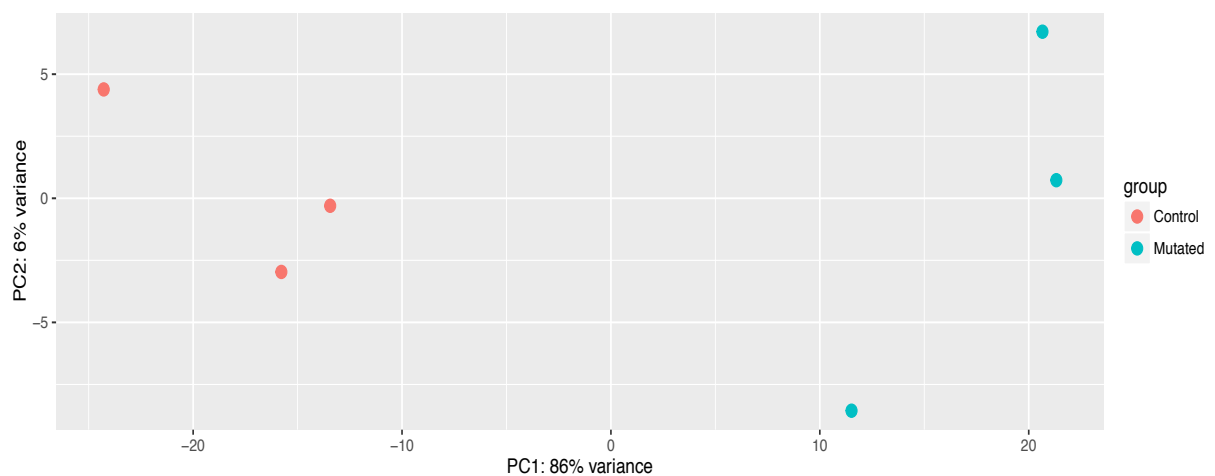


Figure 6.17: PCA plot for all samples.

6.2.4.3: Dispersion Estimation for Differential Gene Expression

As previously discussed in Section 6.6, DESeq uses negative binomial distribution to calculate the dispersion parameter with results from biological variations in order to test for differential expression. Figure 6.18 illustrates the dispersion acquired by the `estimateDispersions` function from DESeq2 package. The black dots are the estimated dispersion for each gene, the red line presents the fitted line which is derived from a generalised linear model. The black dots are then shrunk towards the fitted line to form the blue dots, which are the final estimates of dispersion for each gene. The black dots that are surrounded by blue circles present the dispersion outliers that refer to genes which have very high dispersion estimates. The final estimates that are shown in blue are then used for hypothesis testing [284].

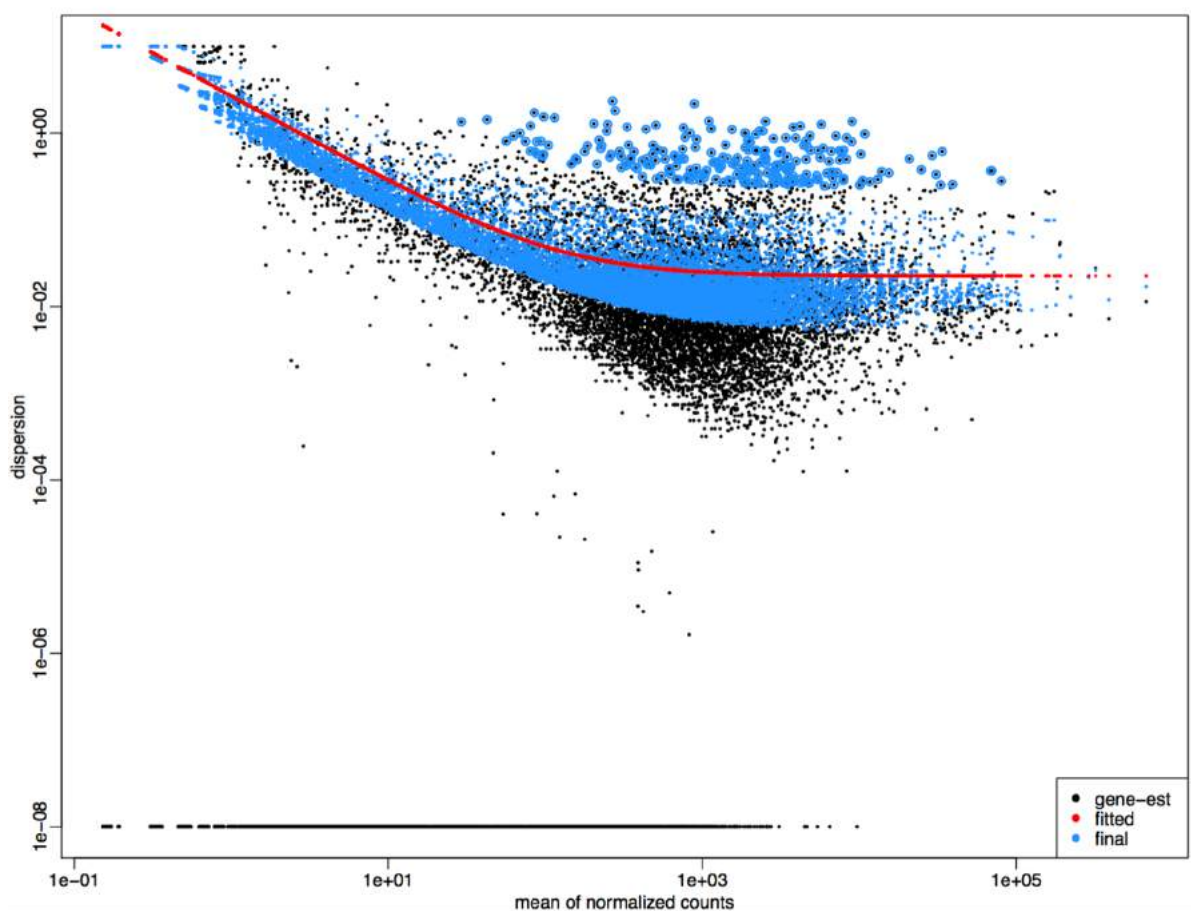


Figure 6.18: Dispersion estimates versus the mean normalised count from DESeq2.

6.2.4.4: Differential Gene Expression Test

A Wald test is used in order to check for differentially expressed genes. A function called `nbinomWaldTest` in the `DESeq2` package uses the dispersion estimates and the calculated size factors to test for the significance of the coefficient in a negative binomial generalised linear model. Then the results of this test can be extracted by the `results` function, in which the Benjamini-Hochberg method is used to return the adjusted p-value. Using the above functions, and filtering out those genes with an adjusted p-value greater than 0.1, 4591 genes were identified as being differentially expressed across two conditions. A snapshot of the results is shown in Figure 6.19.

```
> Results
log2 fold change (MAP): condition Mutated vs Control
Wald test p-value: condition Mutated vs Control
DataFrame with 15682 rows and 6 columns
```

	baseMean	log2FoldChange	lfcSE	stat	pvalue	padj
	<numeric>	<numeric>	<numeric>	<numeric>	<numeric>	<numeric>
FBgn0000003	146.0969	-0.27566118	0.5026209	-0.5484475	0.583384677	0.72467513
FBgn0000008	926.7929	0.34482564	0.1262640	2.7309892	0.006314455	0.02246643
FBgn0000014	1486.0283	-0.02908645	0.1662619	-0.1749435	0.861124023	0.92034840
FBgn0000015	792.9957	-0.12241881	0.1512252	-0.8095134	0.418219881	0.57795692
FBgn0000017	930.7478	0.33858265	0.1490815	2.2711249	0.023139419	0.06437109

Figure 6.19: Results of DESeq2.

One of the useful methods for visualising the results is an MA plot [286]. The X-axis of an MA plot shows the mean normalised counts, and the Y-axis represents the log2 fold changes in normal vs mutated samples (see Figure 6.20). Each dot in the MA plot represents a gene, and those shown in red are identified as differentially expressed genes using an adjusted p-value of 0.1 as the threshold. It shows that genes with lower mean normalised counts that present high variability are accounted for using a shrinkage method to prevent those genes from dominating the results.

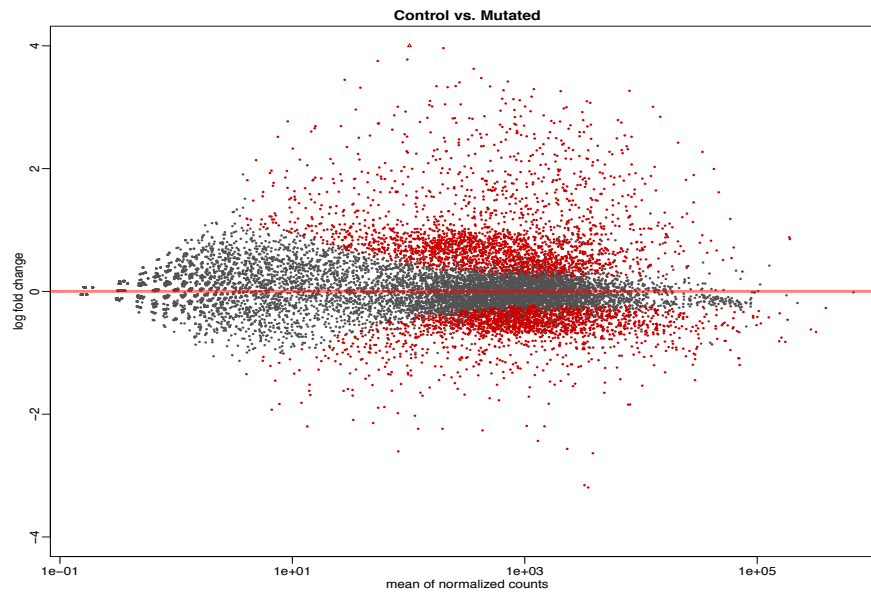


Figure 6.20: MA plot of results using adjusted p-value > 0.1 .

In order to take a more conservative approach to testing for differentially expressed genes, we select those genes that have log₂ fold changes of at least double or half of that between two conditions, and then filter the results based on an adjusted p-value of 0.1. Using this approach, 186 genes were identified as differentially expressed genes and are depicted in the MA plot in Figure 6.21.

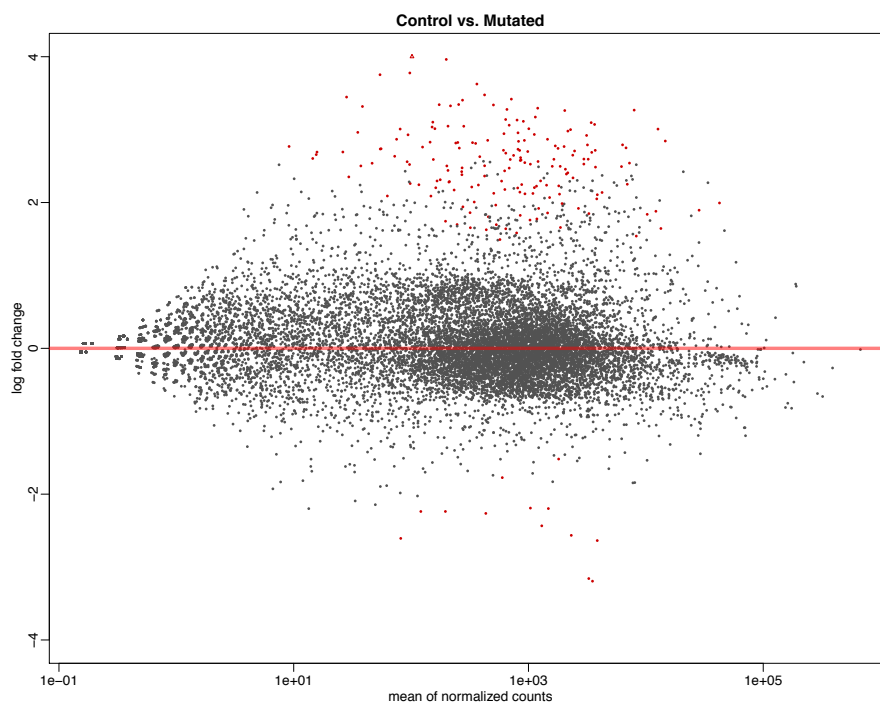


Figure 6.21: MA plot of results using adjusted p-value > 0.1 and log₂ fold changes of at least double or half.

Another useful way to visualise differentially expressed genes is a heat map. To this end, first differentially expressed genes from the previous step (186 genes) should be sorted based on their adjusted p-value. Since homoscedastic data should be used in order to have a valid statistical approach to calculate the distance between the genes, the data from the rlog transformation step should be utilised. By having the gene names form the sorted genes, the relevant dataset is extracted from the rlog matrix. Using an R/Bioconductor package called pheatmap for the top 25 differentially expressed genes, the heat map in Figure 6.22 is acquired. A clear trend can be seen from the heat map, in which the genes on the upper part of the heat map are highly expressed in control samples, while those genes in the lower part are highly expressed in the mutated samples.

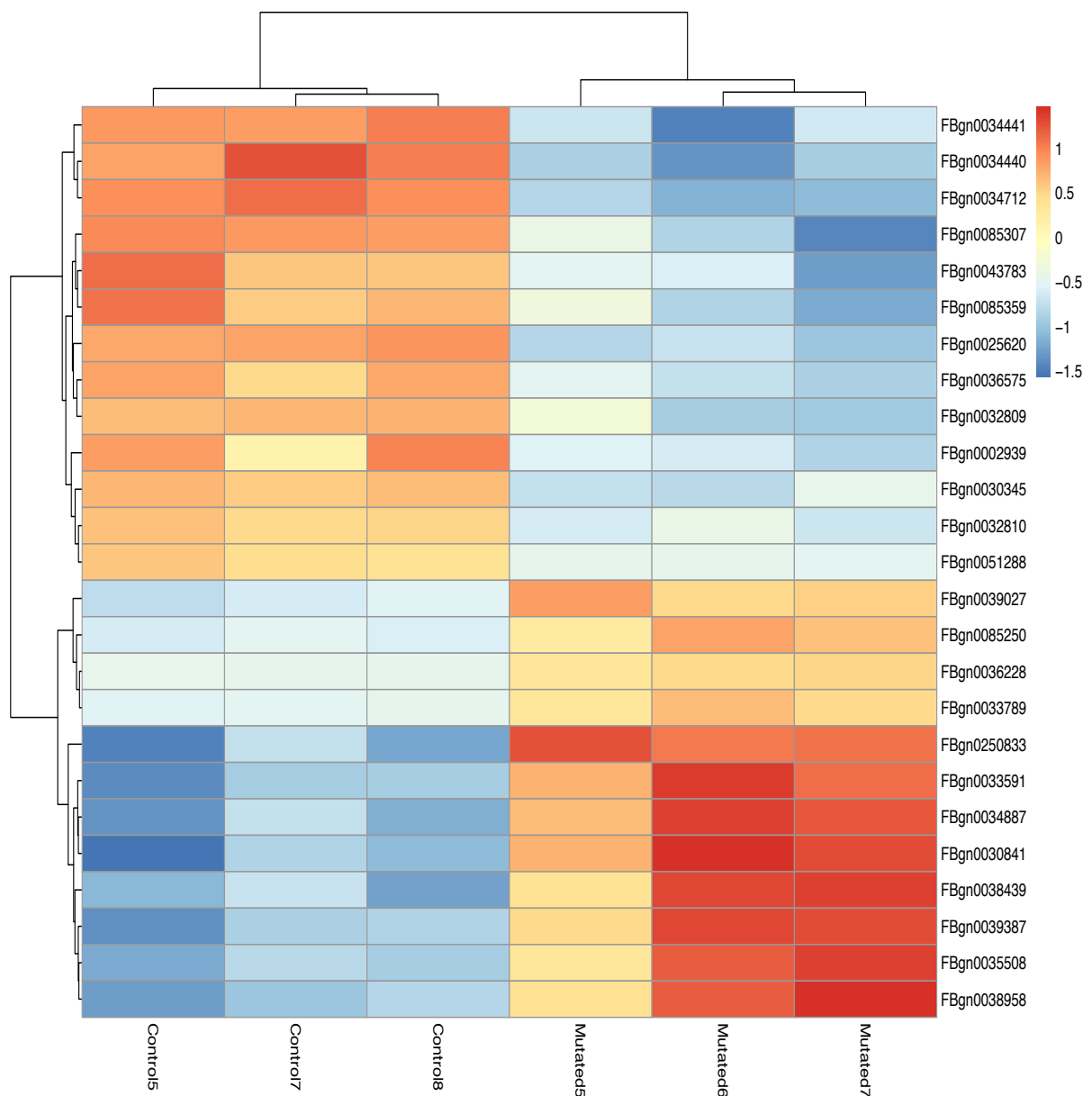


Figure 6.22: Heat map of top 25 differentially expressed genes.

The information on the used packages and their versions is listed in Figure 6.23: R/Bioconductor session information for differential gene expression.

```
> sessionInfo()
R version 3.3.0 (2016-05-03)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.11.6 (El Capitan)

locale:
[1] en_GB.UTF-8/en_GB.UTF-8/en_GB.UTF-8/C/en_GB.UTF-8/en_GB.UTF-8

attached base packages:
[1] stats      graphics  grDevices  utils      datasets  methods   base

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.5           RColorBrewer_1.1-2    GenomeInfoDb_1.8.1     plyr_1.8.4
 [5] XVector_0.12.0        GenomicFeatures_1.24.3 bitops_1.0-6           tools_3.3.0
 [9] zlibbioc_1.18.0       rpart_4.1-10          biomaRt_2.28.0         annotate_1.50.0
[13] RSQLite_1.0.0         gtable_0.2.0          lattice_0.20-33        Matrix_1.2-6
[17] DBI_0.4-1             parallel_3.3.0         gridExtra_2.2.1        genefilter_1.54.2
[21] cluster_2.0.4         rtracklayer_1.32.1     Biobstrings_2.40.2     S4Vectors_0.10.1
[25] IRanges_2.6.1         locfit_1.5-9.1         nnet_7.3-12            stats4_3.3.0
[29] grid_3.3.0            data.table_1.9.6       Biobase_2.32.0         AnnotationDbi_1.34.3
[33] XML_3.98-1.4          survival_2.39-5        BiocParallel_1.6.2     foreign_0.8-66
[37] latticeExtra_0.6-28   Formula_1.2-1          geneplotter_1.50.0     DESeq2_1.12.3
[41] ggplot2_2.1.0         Rsamtools_1.24.0       Hmisc_3.17-4           scales_0.4.0
[45] GenomicAlignments_1.8.3 BiocGenerics_0.18.0    GenomicRanges_1.24.2   splines_3.3.0
[49] SummarizedExperiment_1.2.3 xtable_1.8-2          colorspace_1.2-6       acepack_1.3-3.3
[53] RCurl_1.95-4.8        munsell_0.4.3          chron_2.3-47
```

Figure 6.23: R/Bioconductor session information for differential gene expression.

6.2.5: Differential Exon Usage Analysis

After aligning the RNA-Seq short reads similar to those in Section 6.1.3, one can investigate which exons are expressed differently across two conditions (normal vs mutated). As discussed in Section 6.1.7, to create a valid exon model in which an exon is only counted once, it is required to work with a flattened annotation file (GTF file). However, after creating this file, similar steps to differential gene expression are followed, including counting reads over genes, normalisation, dispersion estimation, and testing for differential exon usage. The R/Bioconductor code written for this analysis can be found in appendix 2.

6.2.5.1: Preparing the Flattened Annotation File

In order to create a flattened file (GFF format) from the annotated file (GTF file), first the relevant GTF file for *Drosophila* (the same version that was used for aligning reads using

STAR) is downloaded from the Ensembl website (Drosophila_melanogaster.BDGP5.76.gtf). Then using Unix terminal and a python file provided by the DEXSeq package (dexseq_prepare_annotation.py), one can easily create the flattened file as shown below:

```
Python DEXSeq/python_scripts/dexseq_prepare_annotation.py --aggregate=no Drosophila_melanogaster.BDGP5.76.gtf
Drosophila_melanogaster.BDGP5.76.gff
```

where the argument *--aggregate* specifies if an exon cannot be assigned to a unique gene, it should be ignored.

6.2.5.2: Counting Reads Over Exon Bins and Creating a DEXSeq Object

The DEXSeq package provides a python file (dexseq_count.py) that uses a HT-Seq functionality (htseq-count) that has been modified to count the number of reads over exons. Using Unix terminal, a .txt formatted file can be created that contains these counts as shown below:

```
DEXSeq/python_scripts/dexseq_count.py --format=bam --paired=yes --stranded=no Drosophila_melanogaster.BDGP5.76.gff
control7.bam control7.txt
```

The above command should be run for each alignment file, separately changing the last two arguments' names to correspond to the sample name under process. Since the aligned files are in BAM format, paired end, and not strand specific, this information is supplied by argument *--format*, *--paired*, and *--strand* respectively. After running the above command six times, each time for different samples, six .txt formatted files are acquired that are used for further analysis after being imported into R/Bioconductor.

By using the DEXSeqDataSetFromHTSeq function from DEXSeq package, the required data frame for DEXSeq is then created. This function includes four arguments that need to be provided. The first argument accepts a character formatted file containing the directory path to count files (.txt files). The second argument is the sample information, which includes 6 rows and 2 columns. Each row corresponds to a sample, the first column is the sample names and the second column contains information on sample conditions, which is similar to the sample information provided in Section 6.2.4.1. The third argument is the design formula that specifies we are looking for exon expression differences based on different conditions. Finally, the last argument is a character formatted file that contains the directory to the GFF files that was created using the python command.

The resulting file from the `DEXSeqDataSetFromHTSeq` function is saved as `DEXSeqDataFrame` which is in the `DEXSeqDataSet` format, and its information can be accessed similarly to that of the `DESeqDataSet` format explained in Section 6.2.4.1. The resulting `DEXSeqDataFrame` file for the six *Drosophila* samples has a dimension of 77026 by 12. The rows of the matrix correspond to the exon IDs, and several rows can be related to a given gene depending on the number of exons for that gene. The first six columns correspond to six samples, each column contains the number of reads assigned to the respective exon ID in a given gene, and we refer to them as the group *A* columns. The next six rows provide information on the sum of reads that are assigned to other exons within the same gene, and we refer to them as the group *B* columns. The `DEXSeq` package compares these two groups in order to identify an exon as being differentially expressed or not. Figure 6.24 illustrates the first 5 rows of this matrix for five exons that originate from two genes.

```
> head( counts(DEXSeqDataFrame), 5 )
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
FBgn00000003:E001      0      1      0      1      0      0      0      0      0      0      0      0
FBgn00000008:E001      3      1      2      3      3      4 1176 1159 1111 1379 1118 1631
FBgn00000008:E002     13      5     10     10      9     22 1166 1155 1103 1372 1112 1613
FBgn00000008:E003     16     10     11     19     13     15 1163 1150 1102 1363 1108 1620
FBgn00000008:E004      8      6      5      5      4      7 1171 1154 1108 1377 1117 1628
```

Figure 6.24: Count table in `DEXSeqDataFrame`.

6.2.5.3: Normalisation of Counts

Similarly to differential gene expression analysis, it is also essential to account for the variation in the sequencing depths across different samples in differential exon usage. Using the `estimateSizeFactors` function of `DEXSeq` on `DEXSeqDataFrame`, the corresponding normalisation factors for samples can be calculated (12 columns as above). The table below gives the information on the estimated size factors for each sample.

Table 6.4: Estimated size factors for each sample using `DEXSeq`.

	Control5	Control7	Control8	Mutated5	Mutated6	Mutated7
Size factors for Group A column	1.07	1.05	0.94	1.01	0.87	1.10
Size factors for Group B column	1.07	1.05	0.94	1.01	0.87	1.10

To find out if the calculated size factors are satisfactory for further analysis, two plots were investigated, including the density of mean counts plot (see Figure 6.25) and the ECDF plot (see Figure 6.26). Two clear distributions can be seen from Figure 6.25 that correspond to both the group A and B columns, and since the line graphs for samples in each group almost overlap, it can be concluded that a satisfactory normalisation was performed. A similar trend is observed in the ECDF plot that provide further evidence to support this conclusion.

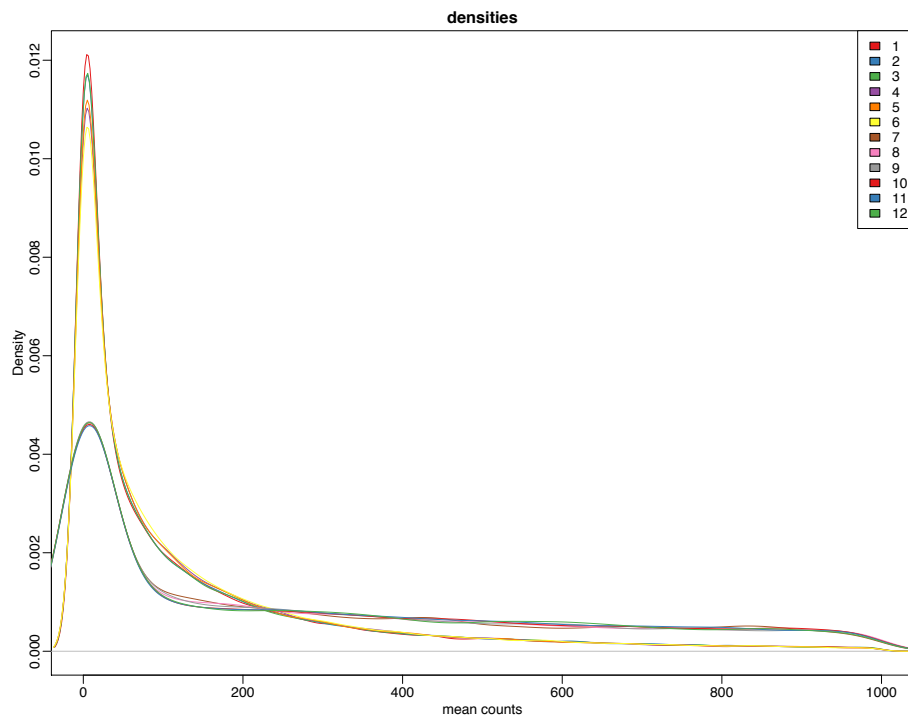


Figure 6.25: Density of mean counts for all samples including group A (1-6) and B (7-12).

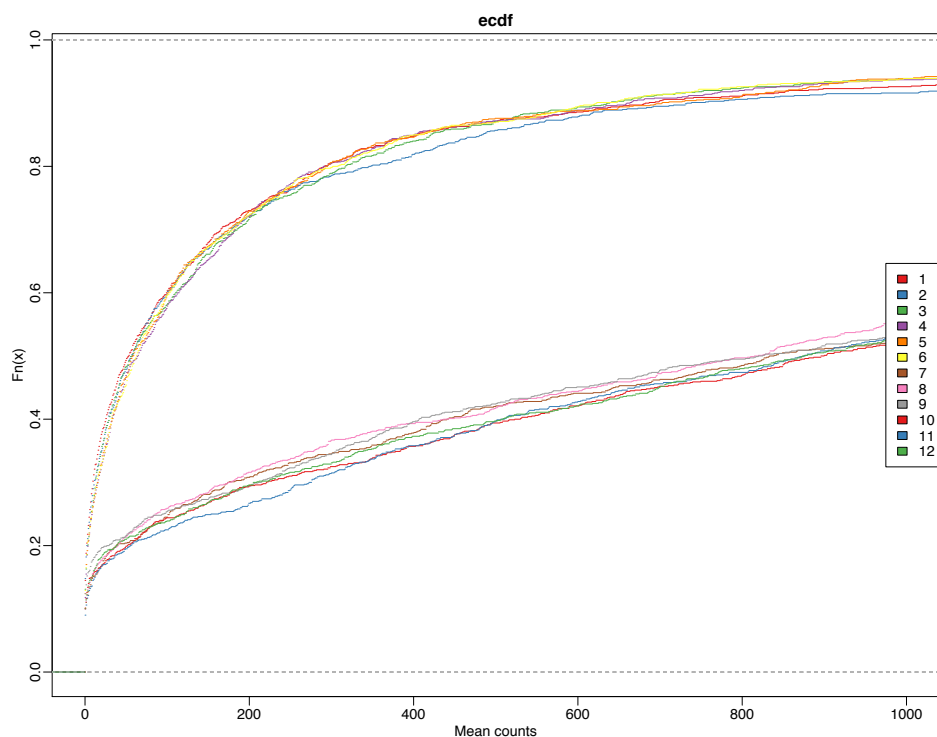


Figure 6.26: Probability of observing a given number of counts for all samples including group A (1-6) and B (7-12).

6.2.5.4: Dispersion Estimation for Differential Exon Usage

For differential exon usage, it is also essential to account for biological variations across samples, so that more interesting variations can be addressed. Figure 6.27 depicts dispersions obtained by the `estimateDispersions` function from the `DEXSeq` package, which is actually the same function as that of `DESeq2`, and the resulting figure can be interpreted as such.

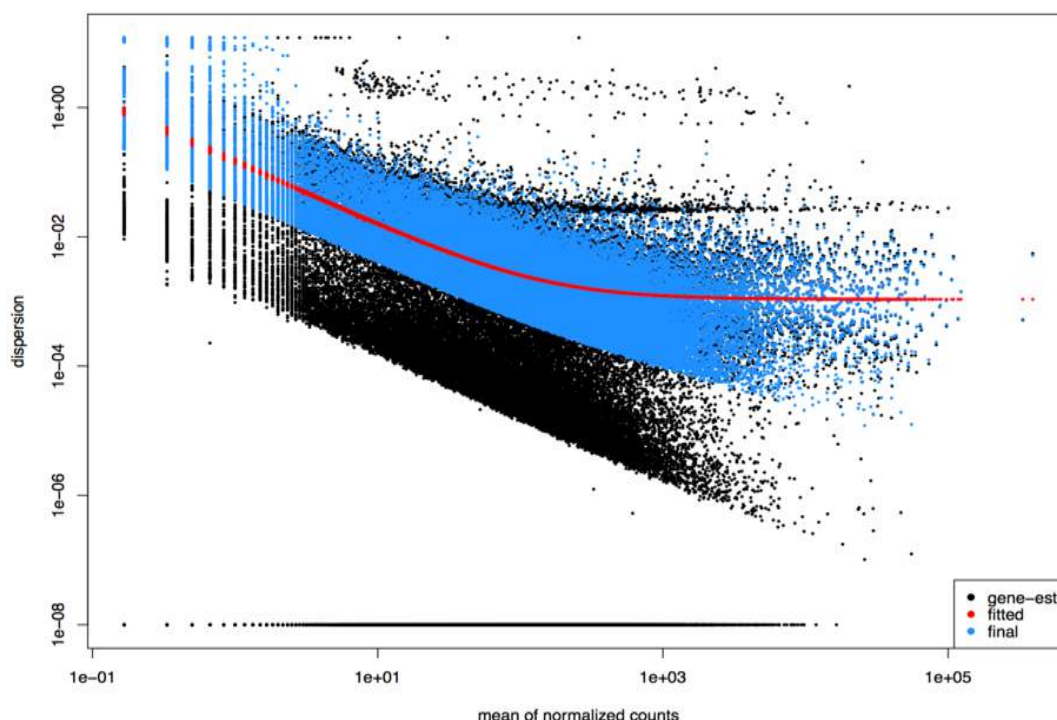


Figure 6.27: Dispersion estimates versus the mean normalised count from DEXSeq.

6.2.5.5: Testing for Differential Usage of Exons

A likelihood ratio test (chi-squared distribution) is then used to test for differential exon usage [272]. This test is performed using the `testForDEU` function from the DEXSeq package, which uses the calculated dispersions and size factors and returns a p-value. Adjusted p-values are also provided using the Benjamini-Hochberg method that can be used for multiple testing. Then the log2 fold changes and the exon usage coefficient are calculated using the `estimateExonFoldChanges` function. The results of these steps are saved as meta data for the DEXSeqDataFrame object, and can be accessed using the DEXSeqResults function. By filtering the data using an adjusted p-value of 0.1, 1053 exons were identified as being differently used across the control and mutated samples that correspond to 622 genes. A snapshot of the results is shown in Figure 6.28.

```
> head(DEXSeq_Results)
```

LRT p-value: full vs reduced

DataFrame with 6 rows and 13 columns

	groupID	featureID	exonBaseMean	dispersion	stat	pvalue	padj	Control	Mutated	log2fold_Mutated_Control
	<character>	<character>	<numeric>	<numeric>	<numeric>	<numeric>	<numeric>	<numeric>	<numeric>	<numeric>
FBgn0000003:E001	FBgn0000003	E001	0.3225461	NA	NA	NA	NA	NA	NA	NA
FBgn0000008:E001	FBgn0000008	E001	2.6473827	0.015090620	0.38894093	0.5328566	NA	2.746774	3.241178	0.23878018
FBgn0000008:E002	FBgn0000008	E002	11.2591219	0.030653687	0.33360444	0.5635443	0.9643206	5.928763	6.529168	0.13916791
FBgn0000008:E003	FBgn0000008	E003	13.8802673	0.003918004	0.06756364	0.7949177	0.9874322	6.808690	7.015312	0.04313006
FBgn0000008:E004	FBgn0000008	E004	5.7157116	0.007088665	1.01194139	0.3144382	0.9122464	4.883672	4.097909	-0.25307837
FBgn0000008:E005	FBgn0000008	E005	73.1126563	0.003139354	1.80972607	0.1785411	0.8398140	15.138571	16.293850	0.10609851

	genomicData	countData	transcripts
	<GRanges>	<matrix>	<list>
FBgn0000003:E001	3R:2648220-2648518:+	0 1 0 ...	#####
FBgn0000008:E001	2R:18024494-18024495:+	3 1 2 ...	#####
FBgn0000008:E002	2R:18024496-18024531:+	13 5 10 ...	#####
FBgn0000008:E003	2R:18024532-18024713:+	16 10 11 ...	#####
FBgn0000008:E004	2R:18024938-18025504:+	8 6 5 ...	#####
FBgn0000008:E005	2R:18025505-18025756:+	78 54 54 ...	#####

Figure 6.28: Results of DEXSeq.

The figure below shows an MA plot in which the red dots represent the exons that are differently used across two conditions and identified as significant using an adjusted p-value of 0.1. It is shown that most of the exons that are identified as significant have a higher number of counts.

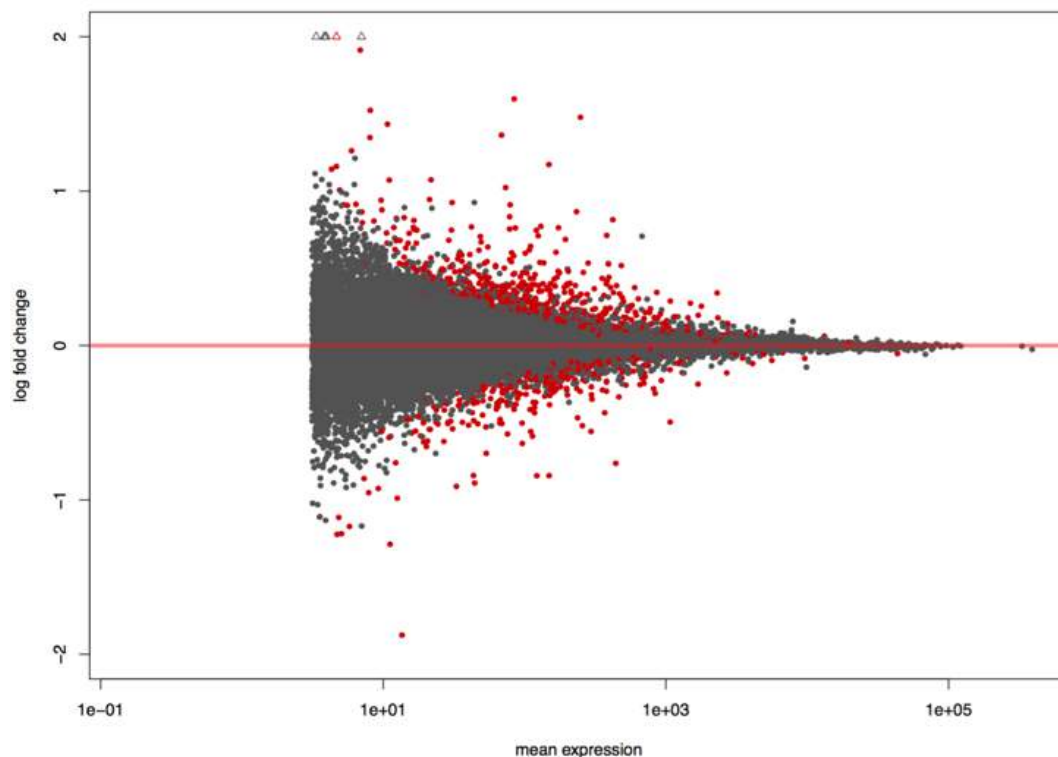


Figure 6.29: MA plot for differential exon usage.

The results of DEXSeq can be visualised using the plotDEXSeq function. Figure 6.30 illustrates the mean expression level for the exons of the FBgn0000382 gene that has the lowest adjusted p-value from the result of the analysis. The transcript models that can be used to visualise isoform expression of this gene are also included.

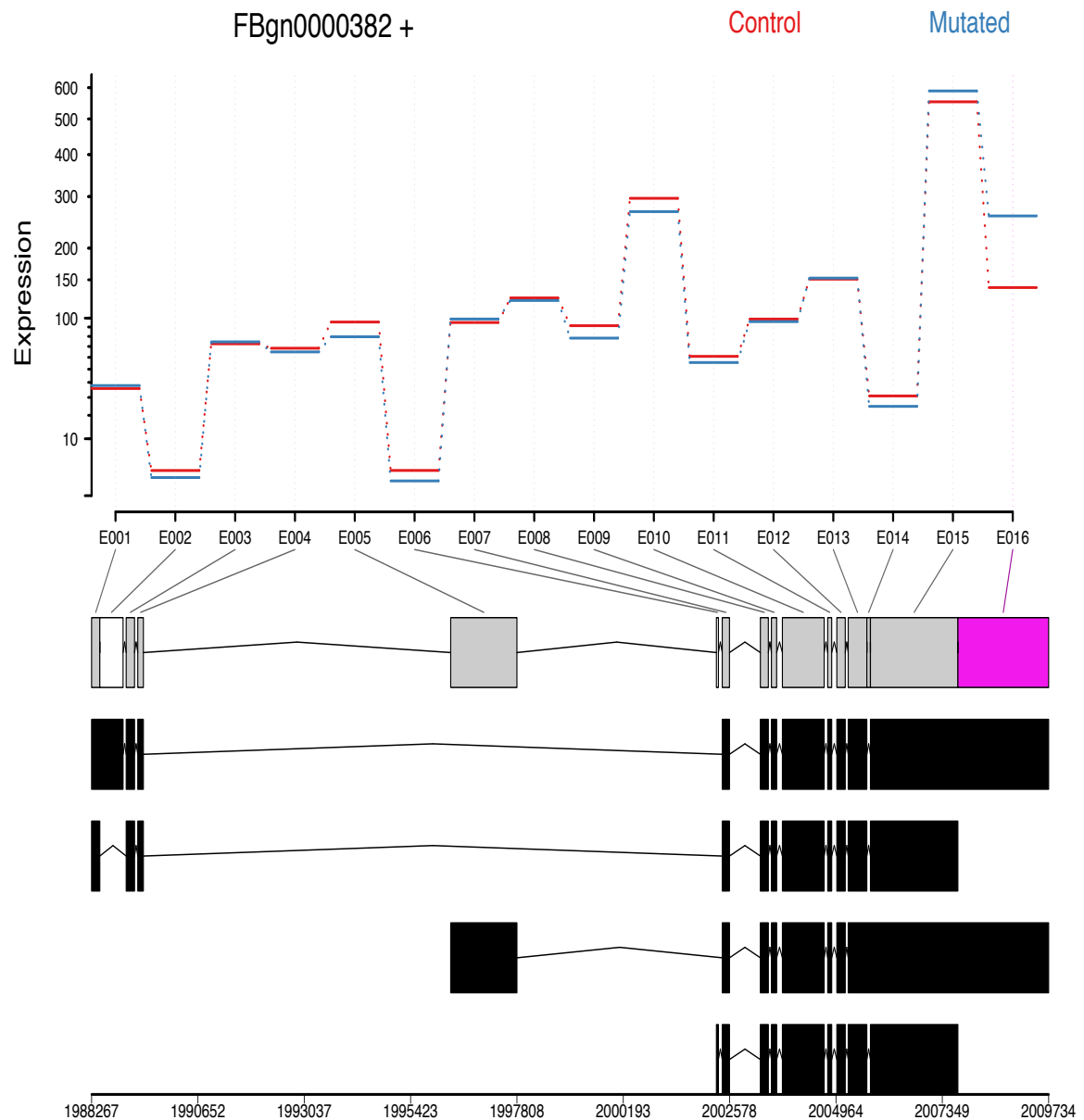


Figure 6.30: Mean expression level for exons of the FBgn0000382 gene.

The information on the used packages and their versions is listed in Figure 6.31.

```
> sessionInfo()
R version 3.3.0 (2016-05-03)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.11.6 (El Capitan)

locale:
[1] en_GB.UTF-8/en_GB.UTF-8/en_GB.UTF-8/C/en_GB.UTF-8/en_GB.UTF-8

attached base packages:
[1] stats4    parallel  stats     graphics  grDevices  utils      datasets  methods   base

other attached packages:
[1] DEXSeq_1.18.4          RColorBrewer_1.1-2      AnnotationDbi_1.34.3
[4] DESeq2_1.12.3          SummarizedExperiment_1.2.3 GenomicRanges_1.24.2
[7] GenomeInfoDb_1.8.1     IRanges_2.6.1           S4Vectors_0.10.1
[10] Biobase_2.32.0         BiocGenerics_0.18.0     BiocParallel_1.6.2

loaded via a namespace (and not attached):
[1] Rcpp_0.12.5            plyr_1.8.4              XVector_0.12.0          GenomicFeatures_1.24.3
[5] bitops_1.0-6           tools_3.3.0             zlibbioc_1.18.0         statmod_1.4.24
[9] rpart_4.1-10           biomaRt_2.28.0          annotate_1.50.0          RSQLite_1.0.0
[13] gtable_0.2.0           lattice_0.20-33         Matrix_1.2-6            DBI_0.4-1
[17] gridExtra_2.2.1        stringr_1.0.0           hwriter_1.3.2           genefilter_1.54.2
[21] rtracklayer_1.32.1     cluster_2.0.4           Biostrings_2.40.2       locfit_1.5-9.1
[25] nnet_7.3-12            grid_3.3.0             data.table_1.9.6        XML_3.98-1.4
[29] survival_2.39-5        foreign_0.8-66          latticeExtra_0.6-28     Formula_1.2-1
[33] magrittr_1.5           geneplotter_1.50.0      ggplot2_2.1.0           Hmisc_3.17-4
[37] Rsamtools_1.24.0       scales_0.4.0            GenomicAlignments_1.8.3 splines_3.3.0
[41] xtable_1.8-2           colorspace_1.2-6        stringi_1.1.1           acepack_1.3-3.3
[45] RCurl_1.95-4.8         munsell_0.4.3          chron_2.3-47
```

Figure 6.31: R/Bioconductor session information for differential exon usage.

6.2.6: Gene Annotation and Biological Relevance of Selected Genes

It is essential to annotate the differentially expressed genes by adding the Entrez ID or gene symbol, which can be done using the AnnotationDbi package. A list of all of the differentially expressed genes (186 genes) including their symbol names can be seen in Appendix 3. First, the biological relevancies of the differentially expressed genes were investigated using the Web tool DAVID (database for annotation, visualisation, and integration discovery) [287]. Three gene ontology (GO) terms including biological process, molecular function, and cellular component were selected.

Table 6.5 gives information on GO terms and the corresponding gene names within each term. With regards to terms for biological processes, ten genes (see Figure 6.32) contribute to the metabolic process of chitin. One of the main substances in the exoskeletons of insects

is chitin [288], and all of the genes that contributed to the metabolic process of chitin are highly expressed, which results in the production of more chitin. Since it is known that AIP positive leads to a bigger body size, enrichment of this term as a result of these 10 genes would suggest that these genes contribute to body size and are linked to CG1847. Another main substance that contributes to the exoskeletons of insects is cuticle [288], which is also shown as an enriched term in the biological process of GO analysis. Ten genes (see Figure 6.33) that are all highly expressed lead to the enrichment of this term in the biological process. In a previous study, it was shown that a mutation of *TwdID* that is in the same family of proteins as *TwdIG*, *TwdIV*, and *TwdIZ* changes body shape in *Drosophila* [289]. These two biological process terms (chitin and cuticle) also appeared in the molecular function of GO analysis with the lowest p-value observed. Proteolysis and the lipid catabolic process are both known to contribute to the breakdown of protein and lipids respectively.

Table 6.5: GO analysis.

GO	Term	genes	P-value
Biological process	metabolic process of chitin	10	9.8E-5
	chitin based cuticle development	10	2.6E-4
	proteolysis	15	1.8E-3
	lipid catabolic process	4	9.4E-2
	Phagocytosis	5	9.4E-1
Cellular component	extracellular region	17	2.3E-6
	extracellular matrix	6	4.7E-3
	integral component of plasma membrane	7	7.8E-3
	extracellular space	9	1.5E-2
Molecular function	chitin binding	11	1.5E-7
	structural constituent of chitin-based cuticle	11	6.4E-7
	serine-type endopeptidase activity	10	2.0E-3
	transferase activity, transferring acyl groups	4	2.1E-3
	lipase activity	4	3.5E-3
	structural constituent of chitin-based larval cuticle	6	4.1E-3
	carbohydrate binding	5	4.8E-3

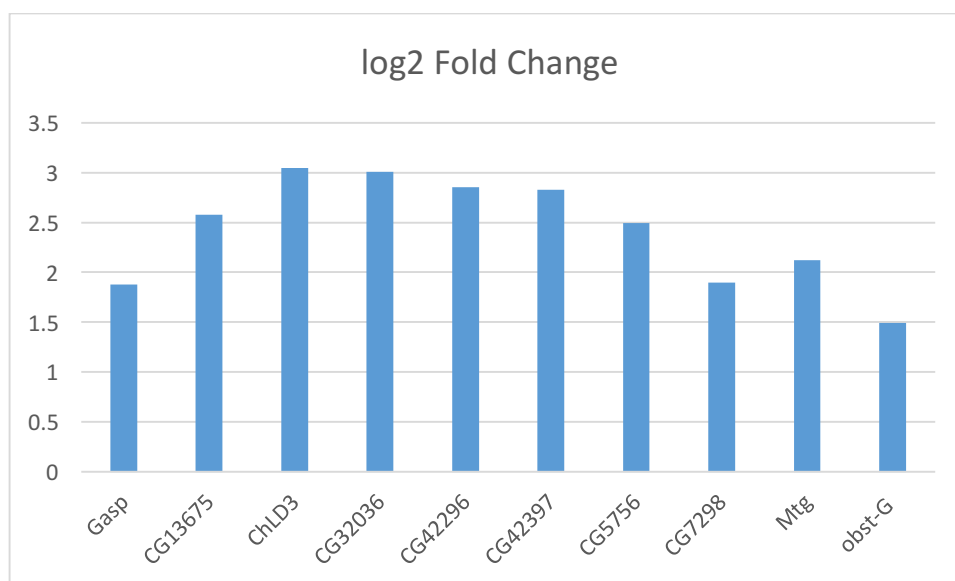


Figure 6.32: Log2 fold change of genes contributing to metabolic process of chitin.

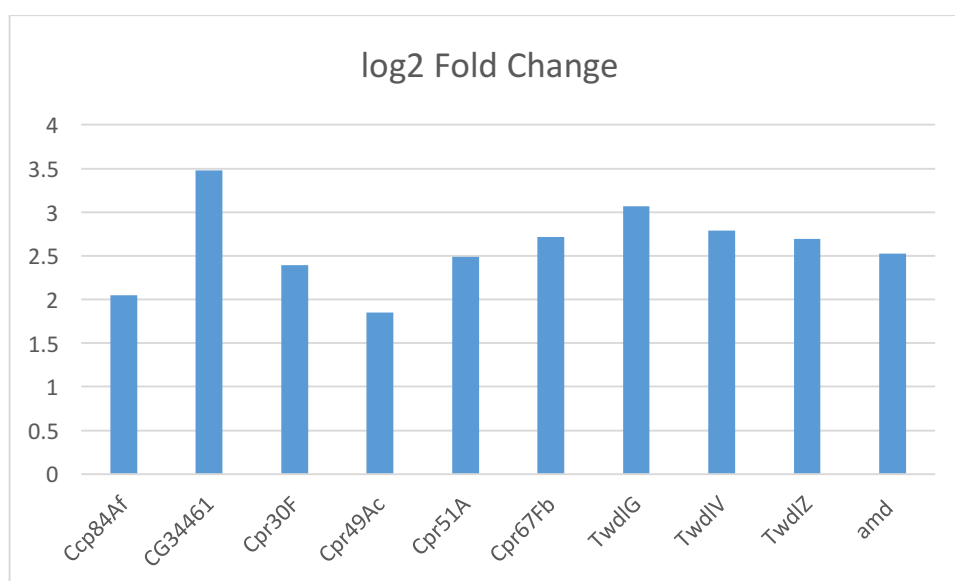


Figure 6.33: Log 2 fold change of genes contributing to chitin-based cuticle development.

Using the functional annotation chart of DAVID for pathway analysis, two terms from the KEGG pathway [290] were enriched, including glycerolipid metabolism and folate biosynthesis. Research suggests that glycerolipid metabolism can control cell growth and cellular function [291]. Three genes including CG4582, CG5665, and CG6753 were identified to contribute to this pathway. Furthermore, two genes, CG3264 and CG8147 led to

the enrichment of the folate biosynthesis pathway. Folate is known to be important in the formation of new cells and their maintenances [292].

6.2.7: Classification

In order to apply a similar methodology to that of microarray for classification purpose of RNA-Seq data, it is essential to normalise RNA-Seq data as explained in Section 6.2.4.2. Since it was observed that the rlog transformation performed better compared to other methods like vst, in this pipeline the rlog of the count is used for classification purposes. Once the transformed count matrix is acquired, those features that have zero counts are removed. The resulting matrix can then be treated like microarray gene expression, and the standard procedure for classification analysis such as feature selection, designing a classifier, and classifier validation can be followed.

Our proposed method for feature selection and classification in Chapter 5 (MRMR-COA-HS) was utilised for RNA-Seq classification. Figure 6.34 illustrates the steps to be performed for the proposed method. In brief, RNA-Seq data is first discretised into nine states (see Section 5.3). Then, the top 100 features were selected using the MRMR filter method of feature selection (see Section 5.4) in the first stage of selection. This was done in order to reduce the computational time for the second stage of selection. In the second stage, the proposed COA-HS was utilised in a wrapper setup with the SVM classifier to minimise the number of selected features, while maintaining a high accuracy for classification. A cost function similar to that in Section 5.5 was used for COA-HS, and the LOOCV model of validation was used to assess the performance of the SVM classifier.

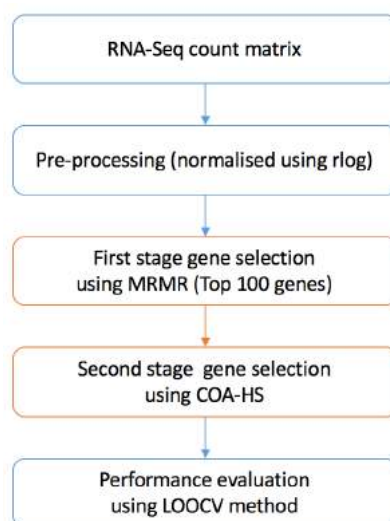


Figure 6.34: Schematic of the general methodology for RNA-Seq classification.

Following the proposed method (See Figure 6.34), 100 features were selected in the first stage of gene selection. In the second stage of gene selection these 100 features were used as input for COA-HS algorithm that is wrapper method and uses SVM to evaluate the features in terms of their power to discriminate between two experimental conditions. After 100 iterations of COA-HS, six genes were selected which led to 100% classification accuracy. The selected genes include CG9021, CG14960, TwdlG, Osi24, CG6741, and CG9154. Figure 6.35 illustrates comparative performance assessments of the SVM classifier for the selected six features, and those 100 features that were selected after the first stage of selection using MRMR. It can be seen that when 100 genes were used for classification purposes, an accuracy of 95.8% was achieved compared to 100% SVM classifier accuracy using the selected six genes as its input.

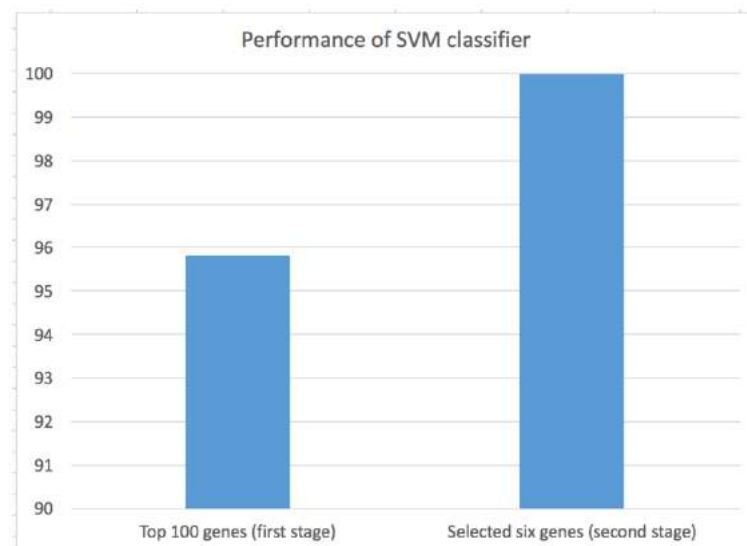


Figure 6.35: Accuracy of SVM classifier.

As mention in Section 6.2.4.4, when DESeq Bioconductor package was used to determine differentially expressed genes, 186 genes were selected based on an adjusted p-value of 0.1 and log2 fold changes of at least double or half of that between two conditions. In the classification analysis using MRMR-COA-HS when normalised counts were used, in fact four genes (CG9021, CG14960, TwdlG, and Osi24) out of the six selected genes by the classification method were also among the 186 genes selected by DESeq Bioconductor package. CG6741 and CG9154 genes that were selected by the classification method and were not found to be differentially expressed by DESeq, were then investigated to determine whether they have biological relevance for the dataset under investigation. It was found that the CG9154 gene is a protein coding gene, and its biological process is the positive regulation of transcription

from the RNA polymerase II promoter. The CG6741 gene is also a protein coding gene, and its biological process is involved with compound eye development. To date, there is limited amount of information available in the literature in regard to the selected genes. However, further exploration in the roles of these genes could shed more light on the function of these genes.

6.2.8: Summary

In this chapter, first an overview of RNA-Seq data analysis was given. Then a state-of-the-art pipeline for RNA-Seq analysis was investigated.

In respect to the overview of RNA-Seq data analysis, different steps required for successful RNA-Seq analysis were explored (see Figure 6.6). First, the importance of experimental consideration in the design of RNA-Seq was pointed out with regards to sequencing depth and number of replicates. Then the sources of possible contaminations in such experiments were explored, including technical and biological contaminations, and how one can eliminate such contaminations as a pre-processing step towards a successful RNA-Seq downstream analysis. It was discussed that the first step after pre-processing is aligning the short reads, either to reference transcripts or a reference genome. The aligner software should be spliced-aware if short reads are mapped to a reference genome, to account for exon-exon junctions. Examples of such aligners are STAR and TopHat. The aligner software usually creates a BAM file, which contains the genomic coordinates that reads are mapped to, and from this file a count matrix is then formed that summarises the number of reads for each genomic feature depending on the objective of the study. Several software was introduced to create the count table. Then different normalisation methods including RPKM were explored that aid in accounting for gene length and library size biases. In Section 6.1.6, statistical methods for modelling raw counts and estimating overdispersion that present in RNA-Seq data due to biological variation were investigated including the negative binomial model. Finally, the concept of differential expression at gene and transcript levels were examined, and some of the well-known software for such analysis were identified.

In respect to the state-of-the-art RNA-Seq analysis pipeline, details of this pipeline were outlined (see Figure 6.7). To perform the analysis, RNA-Seq data from *Drosophila*, including three normal and three AIP positive samples were used. Since there are many different R/Bioconductor packages, a state-of-the-art pipeline was implemented to perform the RNA-Seq analysis for differential gene expression, differential exon usage, sample classification, annotation, and pathway analysis. Initially all samples were quality checked, and when required, pre-processing steps were performed to eliminate noise and low quality RNA-Seq

reads. Then the data was mapped to the *Drosophila* genome using the STAR aligner. These steps were carried out using OSX terminal. Afterwards, the data was imported into the R/Bioconductor software, and the initial differential gene expression analysis was carried out. Several steps were required for this analysis, including counting reads, normalisation, dispersion estimation, and differential gene expression tests, all of which were explored in detail. As a result, 186 genes were identified as differentially expressed. Furthermore, the relevancies of these genes were investigated using gene annotation and tools like DAVID. It was observed that the selected genes play an active role in biological processes, such as chitin and cuticle development. It is noted that both of these substances are related to the main structure for *Drosophila* body size. It is also known that cases that are AIP positive usually lead to bigger body sizes. Therefore, the selected genes could be used as a biomarker for such cases. Furthermore, a differential exon usage analysis was performed to identify any exon that is expressed differently across two conditions. Similar steps to that for differential gene expression were performed. It was observed that 1053 exons were differently used across control and mutated samples that correspond to 622 genes, with gene CG3954 (FBgn0000382) having the smallest p-value. Finally, the use of machine learning for RNA-Seq data was investigated. To this end, the proposed method for microarray data in Chapter 4 was implemented for RNA-Seq data classification. Initially, the count matrix was normalised and transformed using the rlog method, so that the data could have the characteristics of those in homoscedastic data. The transformed count matrix then underwent a two-stage feature selection in order to select the most informative features. As a result of classification, six genes were identified that achieve 100% classification accuracy for the SVM classifier. Four out of six genes were previously identified as differentially expressed. However, features including CG6741 and CG9154 were not observed in the previous analysis.

Chapter 7: Conclusions and Future Research

Recent advances in gene expression have paved the way for investigating it on a genome-wide scale. This has been possible with the help of technologies such as microarray and next generation sequencing, which in principle are very different one from another. This is because the resulting datasets from each technology require different approaches for a successful analysis. This thesis presented an investigation into the analysis of gene expression from both technologies, and provided new methods towards a more successful analysis for multi-category diseases.

7.1: Analysis of Microarray Data

With regards to microarray technology, cancer classification is of the utmost importance. It improves personalised medicine by providing information for better treatment decisions by doctors. However, highly accurate disease classification remains challenging due to the curse of dimensionality in these datasets. Therefore, one of the main objectives of this study was to design solutions to enhance the classification accuracy of microarray data.

Different steps were required to achieve a high classification accuracy, such as gene selection, clustering, and different classifiers, which were explored. It was noted in Chapter 3 that in order to have a successful analysis, several aspects should be considered prior to the analysis such as the design of the microarray experiment and pre-processing, in order to remove systematic errors that present in microarray data. A detailed investigation into unsupervised methods such as K-mean, C-mean, hierarchical clustering, SOM, Bi-CoPaM, and UNCLES were carried out, and the importance of these methods in the visualisation and interpretation of experimental results was pointed out. With regards to the gene selection step, different methods such as filter and wrapper approaches were examined, and two classifiers, SVM and the MLP artificial neural network were studied. Several cancer datasets

including leukaemia, prostate cancer, and lymphoma were used to test the proposed methods for enhancing classification performances.

Several original contributions have been made to this thesis that enhanced the accuracy of cancer classification. For example, a novel gene selection method, in which optimisation based clustering algorithms were utilised in order to cluster microarray data prior to gene selection was developed. This method incorporated a shuffling technique to choose the most informative genes and consequently led to a better classification performance. In this method, a new optimisation algorithm, COA-GA, was also proposed, for which a comparative performance assessment with other optimisation algorithms suggested that the proposed algorithm outperforms other optimisation algorithms such as PSO, GA, and COA in reaching a better minimum in fewer iterations. This ultimately resulted in better classification. However, it was noted that traditional clustering methods such as K-means, C-means, and hierarchical may not have any effects on the classification performance. Furthermore, from a comparative analysis between SVM and MLP, it was observed that the SVM classifier performs better than MLP for microarray cancer classification.

Another method with regard to microarray technology was developed that is called MRMR-COA-HS, which selects the most informative genes in two stages and provides high classification accuracy for cancer datasets under investigation. In the proposed method, initially the most relevant genes were selected using MRMR, which is a filter method, to reduce the computational time for the second stage of the selection process. In the second stage, a novel optimisation algorithm called COA-HS was proposed, and the cost function was designed so that the number of selected genes would be minimised while maximising the accuracy of the SVM classifier. The LOOCV method was used to examine the performance of the proposed method, and the results were compared to other algorithms such as PSO, GA, HS, and COA. Overall, this approach resulted in a high classification accuracy for all optimisation algorithms mentioned above. However, it was observed that COA-HS outperformed other methods, by both achieving a higher classification accuracy for all datasets, and selecting a lower number of genes to achieve its accuracy compared to other optimisation methods. Since each algorithm was ran 20 times, those gene that were selected at least 10 times out of 20 runs where then further investigated and found to be biologically relevant to each cancer dataset.

This part of the thesis provides new approaches that enhance prognosis and classification of cancer using microarray data that provides reliable classification results, which can lead to more informed decisions by doctors.

7.2: Analysis of RNA-Seq Data

With regards to next generation sequencing (RNA-Seq), the required primary analysis for successful downstream analysis includes several steps such as pre-processing, alignment of short reads to a reference genome, creation of a count table, and normalisation. However, depending on the downstream analysis, the statistical modelling of the count table and normalisation can differ. Since the introduction of RNA-Seq, the preferred platform for the analysis of such a dataset has been R/Bioconductor, and therefore numerous packages have been proposed to aid in a successful analysis. This has led to an overwhelming number of choices that one can opt for, and many studies have proposed pipelines to use specific software to perform such analyses from start to downstream analysis of choice. However, due to advances in statistical methods that applied to RNA-Seq, these pipelines have undergone several changes. This thesis investigated a state-of-the-art pipeline that uses more cited, recently developed software, and can be used for different steps towards downstream analysis, such as differential gene expression and differential exon usage. Nevertheless, there has not been enough research to apply classification for RNA-Seq thus far, as the focal point for this dataset is finding the features, including genes, exons, and isoforms that are being used differently across different conditions. Therefore, as a part of the proposed pipeline, the classification approach that was used for two-stage gene selection with microarray was utilised to pave the way for using statistical methods from microarray in next generation sequencing for classification purposes.

To investigate this pipeline, RNA-Seq data from AIP deficient *Drosophila* that was produced in house at Queen Mary University of London was used. Initially, a differential gene expression analysis was performed, and important steps including counting reads, normalisation, dispersion estimation, and differential gene expression tests were investigated, and the required packages for these steps were pointed out. As a result of the differential gene expression analysis, 186 genes were identified as differentially expressed. By examining the functions of the differentially expressed genes, it was discovered that these genes are essential in biological processes such as chitin and cuticle development, both of which are important factors for *Drosophila*'s body size. Since AIP-deficient cases can lead to a bigger body size, the selected differentially expressed genes were deemed to play a direct role in the case study under investigation, and can be used as biomarkers.

Differential exon usage was then investigated to provide information for alternative splicing, in which similar steps to that for differential gene expression were identified to be important for a successful analysis, while the differences in modelling count were also pointed

out. As a result of this analysis, 622 genes were identified to have exons that are differently expressed across control and mutated samples (1053 exons).

Finally, the classification was successfully performed for RNA-Seq data. The count table was normalised and then treated as a microarray matrix for classification. For classification purposes, the MRMR-COA-HS method that was proposed in Chapter 4 for microarray data was used. As a result, six genes were selected that led to 100% classification accuracy for SVM. Two out of six of the selected genes by MRMR-COA-HS were not found to be differentially expressed when performing differential gene analysis.

7.3: Suggestions for Future Work

Future work for this research can be divided into two parts. The first part concerns microarray data analysis, and the limitations of the proposed methods being that they are designed for a two-class classification task. However, this can be expanded on for multi-class classifications in number of ways. For instance, a library for SVM is proposed to achieve this objective that is known as LIBSVM [293], and can be used instead of simple SVM. The proposed methods for classification in Chapter 4 and Chapter 5 can be used for extra microarray datasets to further validate these methods, and upon successful validation, these methods could be used as a benchmark for cancer classification. The proposed optimisation algorithms can be used for other optimisation-based problems too, as they outperform other algorithms such as GA, PSO, HS, and COA at achieving a better minimum in fewer iterations. This can reduce the computational time significantly, while providing better results at the same time. Finally, gene selection also plays an important role in achieving high classification accuracy. Therefore, it is worth investigating new feature selection methods that have recently been proposed for other scientific fields like text classification, for the purpose of cancer classification.

The second part relates to RNA-Seq data. As mentioned in Section 6.2.7, once the count table is normalised appropriately, statistical approaches that used for microarray data can be applied in a similar fashion. Therefore, as a future study, one could investigate different clustering methods to identify hidden patterns within RNA-Seq data that could not be observed with other analyses. Furthermore, differential isoform analysis has recently attracted many researchers, and the objective is to observe which isoforms are expressed differently across different experimental conditions. The most cited tool for differential isoform analysis to date is cufflink [264], which would be a good starting point for such analysis.

Appendix 1: R-code for differential gene expression analysis

```
#TO USE PARALLEL COMPUTING
library("BiocParallel")
register(MulticoreParam(5))
#####1. READING DATA INTO R#####
#====dir to datasets
dir="/Users/Main-Data"
#====dirs to bam files
AlignedFiles <- list.files(dir, ".bam$", full.names = TRUE)
#====dir to gtf file
GTFFile <- file.path(dir, "Drosophila_melanogaster.BDGP5.76.gtf")
#====read in bamfiles by Rsamtools
library(Rsamtools)
BAMFileList <- BamFileList(AlignedFiles,yieldSize=10^5)
#==== create sample table
SampleInfo = data.frame(
  row.names = c("Control5","Control7","Control8","Mutated5","Mutated6","Mutated7"),
  condition = c("Control","Control","Control","Mutated","Mutated","Mutated"))

#####2. COUNT THE READS #####
library(GenomicFeatures)
TxDbFromGFF <- makeTxDbFromGFF(GTFFile, format="gtf")
ExonByGenes <- exonsBy(TxDbFromGFF, by="gene")
length(ExonByGenes)
summary(elementNROWS(ExonByGenes))
setSessionTimeLimit(cpu = Inf, elapsed = Inf)
library(GenomicAlignments)
RangedSummarizedExperiment <- summarizeOverlaps(ExonByGenes, BAMFileList,
  mode="Union",
  singleEnd=FALSE,
  ignore.strand=TRUE,
  fragments=TRUE)

###add column data
colData(RangedSummarizedExperiment) <- DataFrame(SampleInfo)
## Visualizing sample-sample distances
plot(assay(RangedSummarizedExperiment)[,2:3])
#####3 CREATE OBJECT FOR DESEQ2 #####
### Creating a DESeqDataSet object
library(DESeq2)
DESeqDataFrame <- DESeqDataSet(RangedSummarizedExperiment, design= ~ condition)
```

```

#####4 NORMALIZATION#####

DESeqDataFrame <- estimateSizeFactors(DESeqDataFrame)
sizeFactors(DESeqDataFrame)
colSums(counts(DESeqDataFrame))
library(geneplotter)
multidensity( counts(DESeqDataFrame, normalized = T),
              xlab="mean counts", xlim=c(0, 1000))

multiecdf( counts(DESeqDataFrame, normalized = T),
           xlab="Mean counts", xlim=c(0, 1000))
#####exploratory data analysis
loggeomeans <- rowMeans(log(counts(DESeqDataFrame)))
hist(log(counts(DESeqDataFrame)[,1]) - loggeomeans,
     col="grey", main="", xlab="", breaks=40)
log.norm.counts <- log2(counts(DESeqDataFrame, normalized=TRUE) + 1)
log.norm <- normTransform(DESeqDataFrame)
rs <- rowSums(counts(DESeqDataFrame))
mypar(1,1)
# not normalised
boxplot(log2(counts(DESeqDataFrame)[rs > 0,] + 1))
# normalised
boxplot(log.norm.counts[rs > 0,])
plot(log.norm.counts[,2:3])
### rld transformation
rld <- rlog(DESeqDataFrame)
plot(assay(rld)[,2:3])
### vsd transformation
vsd <- varianceStabilizingTransformation(DESeqDataFrame)
plot(assay(vsd)[,2:3])
#The principal components (PCA) plot
plotPCA(rld, intgroup="condition")

##### 5 Differential gene expression#####

DESeqDataFrame <- estimateDispersions(DESeqDataFrame)
plotDispEsts(DESeqDataFrame)
#test for differential analysis
DESeqDataFrame <- nbinomWaldTest(DESeqDataFrame)
DESeq2Results <- results(DESeqDataFrame, pAdjustMethod = "BH")
summary(DESeq2Results)
table(DESeq2Results$padj < 0.1)
DESeq2ResultsFoldChange <- results(DESeqDataFrame, lfcThreshold=1)
table(DESeq2ResultsFoldChange$padj < 0.1)
#####FIND DIFFERENTIALLY EXPRESSED GENES#####
#The top "n" high and low expressed genes by adjpval:
n = 5
#Differential-Expressed-Genes with adjusted p-value <0.1

```

```

DiffExprGenes <- DESeq2ResultsFoldChange[ which(DESeq2ResultsFoldChange$padj < 0.1 ), ]
dim(DiffExprGenes)
#all differentially expressed genes
DiffExprGenesbyPadj<- DiffExprGenes[ order( DiffExprGenes$padj ),]
write.csv( as.data.frame(DiffExprGenesbyPadj), file="Diff-Exp-Genes.csv" )
#sort it by the log2 fold change estimate
DiffExprGenesSortedByfoldChange <- DiffExprGenes[ order( DiffExprGenes$log2FoldChange ), ]
#TOP UP and DOWN:
DiffExprGenesbyPadjFold <- rbind(head(DiffExprGenesSortedByfoldChange,n),tail(DiffExprGenesSortedByfoldChange,n))
DiffExprGenesbyPadjFold
write.csv( as.data.frame(DiffExprGenesbyPadjFold), file="Diff-Exp-Genes-TOP30.csv" )
DiffExprGenesbyPadjFold[c(1:5,(2*n-4):(2*n)), c('baseMean','log2FoldChange','padj')]

#*****6 VISUALISING THE RESULTS *****

#===== Dispersion plot
plotDispEsts(DESeqDataFrame)
dev.off()
#===== MA-plot
plotMA(DESeq2ResultsFoldChange, main='Control vs. Mutated', ylim=c(-4,4))
#===== Plot top gene
# Examine the counts for the top gene
mypar(1,2)
plotCounts(DESeqDataFrame, gene=which.min(DESeq2ResultsFoldChange$padj), intgroup="condition")
#the gene which had lowest expression log-fold0change
plotCounts(DESeqDataFrame, gene=which.min(DESeq2ResultsFoldChange$log2FoldChange), intgroup="condition")
#the gene which had which had highest expression log-fold0change
plotCounts(DESeqDataFrame, gene=which.max(DESeq2ResultsFoldChange$log2FoldChange), intgroup="condition")

#===== Heatmap
library(pheatmap)
DiffExprGenesbyPadj<- DiffExprGenes[ order( DiffExprGenes$padj),]
SortedGenes <- DiffExprGenesbyPadj
topgenes <- head(rownames(SortedGenes),25)
topgenes <- rownames(SortedGenes)
#topgenes <- head(rownames(DiffExprGenesDoubleOrHalf),)
# matrix from rld for the selected genes
mat <- assay(rld)[topgenes,]

#Fourth, we subtract the rowMeans from this matrix to have a uniform plot
mat <- mat - rowMeans(mat)
pheatmap(mat)

```

```

#*****6 RESULTS *****
#===== Export results into CSV
write.csv( as.data.frame(resSort), file="results.csv" )
#===== Annotation
#simply add symbol to the genes
DiffExprGenes$SYMBOL <- mapIds(org.Dm.eg.db,
                              keys=row.names(DiffExprGenes),
                              column="SYMBOL",
                              keytype="FLYBASE",
                              multiVals="first")
#simply add entrez to the genes
DiffExprGenes$ENTREZID <- mapIds(org.Dm.eg.db,
                                keys=row.names(DiffExprGenes),
                                column="ENTREZID",
                                keytype="FLYBASE",
                                multiVals="first")

dim(DiffExprGenes)
write.csv( as.data.frame(DiffExprGenes), file="Annotation.csv" )
getwd()
#===== Session Information
Packages_used_in_this_analysis= session_info()

```

Appendix 2: R-code for differential exon usage

```
#TO USE PARALLEL COMPUTING
multicoreWorkers()
BPPARAM = MulticoreParam(workers=5)

#####1. PREPERATION #####
#####Done in TERMINAL
#####1.1 Preparing the annotation
#following should be run once to create flattened gff from gtf file.

python/Library/Frameworks/R.framework/Versions/3.3/Resources/library/DEXSeq/python_scripts/dexseq_prepare_annotation.py --aggregate=no Drosophila_melanogaster.BDGP5.76.gtf Drosophila_melanogaster.BDGP5.76.gff
#
#####1.2. counting reads
#following should be run for all samples
# gff_file is the output file from perevious command
gff_file=Drosophila_melanogaster.BDGP5.76.gff
#change "bam_file" in sequence with desired bam file for all samples (note that the samples should be sorted by name using samtools) # "samtools sort -n sample.bam sample_SortedByName"
bam_file=Mut_4_sortmerna_STAR_SortedByName.bam
out=$bam_file.dexseq_noaggregate.txt

python/Library/Frameworks/R.framework/Versions/3.3/Resources/library/DEXSeq/python_scripts/dexseq_count.py --format=bam --paired=yes --stranded=no $gff_file $bam_file $out

#####2. READING DATA INTO#####

dir="/Users/Main-Data"
CountFilePaths = list.files(dir, pattern="txt$", full.names=TRUE)
basename(CountFilePaths)
class(CountFilePaths)
flattenedFilePath = list.files(dir, pattern="gff$", full.names=TRUE)
basename(flattenedFilePath)
#sample table
SampleInfo = data.frame(
  row.names = c("Control5","Control7","Control8","Mutated5","Mutated6","Mutated7"),condition =
  c("Control","Control","Control","Mutated","Mutated","Mutated"))
```

```

#*****3. CREATE OBJECT*****

library("DEXSeq")
DEXSeqDataFrame = DEXSeqDataSetFromHTSeq(
    CountFilePaths,
    sampleData=SampleInfo,
    design= ~ sample + exon + condition:exon,
    flattenedfile=flattenedFilePath )

#*****3. NORMALIZATION *****

#measure relative sequencing depth using SizeFactor
DEXSeqDataFrame = estimateSizeFactors( DEXSeqDataFrame )
sizeFactors(DEXSeqDataFrame)
library(geneplotter)
mypar(1,1)
multidensity( counts(DEXSeqDataFrame, normalized = T),
    xlab="mean counts", xlim=c(0, 1000))
multiecdf( counts(DEXSeqDataFrame, normalized = T),
    xlab="Mean counts", xlim=c(0, 1000))
dev.off()

#Dispersion estimation (second line to parallel so quick)
DEXSeqDataFrame = estimateDispersions( DEXSeqDataFrame, BPPARAM=BPPARAM)
plotDispEsts( DEXSeqDataFrame )

DEXSeqDataFrame = testForDEU( DEXSeqDataFrame, BPPARAM=BPPARAM)
DEXSeqDataFrame = estimateExonFoldChanges(DEXSeqDataFrame, fitExpToVar="condition", BPPARAM=BPPARAM)
# results table
DEXSeq_Results = DEXSeqResults( DEXSeqDataFrame )
plotMA(DEXSeq_Results, cex=0.8)

table (DEXSeq_Results$padj < 0.1 )
table(tapply(DEXSeq_Results$padj<0.1,DEXSeq_Results$groupID,any))

DifferentialExons <- DEXSeq_Results[ which(DEXSeq_Results$padj < 0.1 ), ]
dim(DifferentialExons)

#you added next 3 line for extra...
TopGenesSorted <- DifferentialExons[ order( DifferentialExons$padj ), ]
dim(TopGenesSorted)

TopGeneSortedNames <- rownames(TopGenesSorted)
TopGeneSortedNames

write.csv( as.data.frame(TopGenesSorted), file="different-exon-usage-ordered.csv" )

```

```

#=====4 VISUALIZATION=====
head(TopGeneSortedNames)
mypar(1,1)
#draw the fitted expression levels of each of the exons of gene FBgn0010909 for each condotion
plotDEXSeq( DEXSeq_Results, "FBgn0000382", legend=TRUE, cex.axis=1.2, cex=1.3, lwd=2 )

# visualize the transcript models, which can be useful for putting differential exon usage results into the context of
isoform expression.
plotDEXSeq( DEXSeq_Results, "FBgn0000382", displayTranscripts=TRUE, legend=TRUE, cex.axis=1.2, cex=1.3, lwd=2 )

#the count values from the individual samples. The counts are normalized by dividing them by the size factors
plotDEXSeq( DEXSeq_Results, "FBgn0000382", expression=FALSE, norCounts=TRUE, legend=TRUE, cex.axis=1.2,
cex=1.3, lwd=2 )

#create browsable, detailed overview over all analysis results, allowing a more detailed exploration of the results.
#saved in getwd()
setwd("~/Desktop")
DEXSeqHTML( DEXSeq_Results, FDR=0.1, color=c("#FF000080", "#0000FF80"),BPPARAM=BPPARAM)

#conclude by adding the session information:
sessionInfo()

```


Appendix 3: Differentially expressed genes

Flyba se ID	base Mean	log2Fold Change	lfcSE	stat e	pvalu	padj	SYM BOL	ENT REZID
FBgn000075	1038.397315	2.526047508	0.341702061	4.466017866	7.97E-06	0.0010503	amd	35188
FBgn000451	14644.71198	2.843459658	0.323419394	5.69990449	1.20E-08	3.26E-06	ect	44135
FBgn001078	2665.015746	1.919381491	0.241009289	3.814713925	0.000136341	0.013031473	ftz-f1	40045
FBgn001254	2965.431248	2.595988359	0.474081053	3.366488387	0.000761318	0.054784259	ImpE2	38432
FBgn002939	81.62736955	-2.606831679	0.380153678	-4.22679504	2.37E-05	0.002853866	ninaD	326160
FBgn003254	147.5194289	2.088461151	0.270785153	4.019648557	5.83E-05	0.00645829	rib	44855
FBgn003292	875.6831834	2.246260366	0.305565891	4.078532329	4.53E-05	0.005206083	rt	39297
FBgn004577	454.9501129	1.803923997	0.240977466	3.336096153	0.000849638	0.060790365	Pxd	2768671
FBgn004778	3817.224852	2.050250037	0.326177318	3.21987452	0.001282467	0.088229515	Ccp84Af	40820
FBgn005638	814.4345336	2.730871678	0.466614407	3.709426139	0.00020773	0.018774509	slbo	37889
FBgn010357	42393.87984	1.993751662	0.288267337	3.447326615	0.000566164	0.043632318	betaTry	47901
FBgn011236	1454.890547	1.8569522	0.199134459	4.303384785	1.68E-05	0.002132887	ken	37785
FBgn022700	3227.900865	2.594146329	0.225107891	7.081699023	1.42E-12	1.62E-09	Cht4	49815
FBgn023214	246.6377467	1.697836664	0.191694199	3.640364009	0.000272253	0.023310225	edl	37149
FBgn023496	618.2178408	2.944139821	0.493328797	3.940860195	8.12E-05	0.008542682	Lip1	43973
FBgn024366	1121.55349	2.933553265	0.273614267	7.066712148	1.59E-12	1.66E-09	CG11409	31078
FBgn025620	3854.774333	-2.636650132	0.168748745	-9.698739601	3.05E-22	1.91E-18	CG13360	31025
FBgn026077	12177.82717	1.879845452	0.190166412	4.626713218	3.72E-06	0.000516859	Gasp	40745
FBgn029573	75.24284505	2.867661816	0.355492104	5.253736429	1.49E-07	2.79E-05	CG14770	31085

FBgn0	7952.	3.267378	0.467	4.850	1.23E-	0.000	CG14	3128
029644	622331	435	424099	794896	06	197395	421	5
FBgn0	6200.	2.493103	0.327	4.553	5.28E-	0.000	CG14	3128
029646	465874	03	908682	411098	06	726259	423	7
FBgn0	125.1	2.760635	0.409	4.294	1.75E-	0.002	CG17	3128
029647	989681	906	943526	825497	05	189056	959	8
FBgn0	2037.	3.261732	0.335	6.740	1.57E-	1.04E-	CG41	3129
029649	288165	193	527722	820635	11	08	16	1
FBgn0	3458.	2.687863	0.489	3.445	0.000	0.043	CG15	3134
029681	528909	788	809553	959308	569036	632318	239	3
FBgn0	1206.	2.562657	0.289	5.394	6.87E-	1.41E-	CG30	3150
029804	38051	415	667218	664352	08	05	97	3
FBgn0	209.9	2.280402	0.309	4.141	3.45E-	0.004	CG59	3153
029836	607695	057	161463	531881	05	11397	28	9
FBgn0	844.9	2.142365	0.281	4.062	4.86E-	0.005	CG46	3154
029838	113309	834	221154	161814	05	518933	66	1
FBgn0	116.5	2.243672	0.309	4.015	5.94E-	0.006	cyr	3173
030001	196895	082	744248	157956	05	524758		3
FBgn0	196.0	-	0.230	-	7.54E-	1.52E-	CG18	3214
030345	794296	2.237414887	097203	5.377791	08	05	47	4
			95					
FBgn0	7250.	2.540801	0.482	3.193	0.001	0.095	CG11	3236
030541	550331	714	461816	624165	404989	488625	584	3
FBgn0	257.5	2.746819	0.513	3.399	0.000	0.049	CG12	3239
030570	31827	933	866457	365554	675424	746947	540	3
FBgn0	699.8	2.590481	0.465	3.414	0.000	0.048	CG95	3241
030590	078851	204	741849	941575	637957	119632	18	6
FBgn0	822.5	2.839263	0.298	6.161	7.18E-	2.73E-	NA	NA
030591	87806	252	486279	969178	10	07		
FBgn0	3858.	2.106950	0.300	3.688	0.000	0.019	CG90	3244
030617	591576	165	110059	48071	225597	753158	95	7
FBgn0	269.8	2.423634	0.251	5.653	1.57E-	4.04E-	CG13	3267
030798	140113	287	825883	248463	08	06	003	5
FBgn0	714.9	3.418263	0.369	6.546	5.90E-	2.96E-	CG85	3272
030841	641447	815	413724	220839	11	08	68	8
FBgn0	9.131	2.768949	0.508	3.481	0.000	0.039	CG62	3282
030921	19017	651	130653	288998	499007	544707	90	7
FBgn0	506.1	2.126314	0.235	4.777	1.77E-	0.000	CG78	3291
031001	244901	79	749115	599238	06	270882	84	4
FBgn0	1872.	1.657625	0.204	3.214	0.001	0.089	CG10	3312
031178	958243	749	589207	371655	307303	446701	918	3
FBgn0	746.7	2.646407	0.316	5.198	2.00E-	3.64E-	CG28	3369
031646	619245	01	681037	944104	07	05	37	6
FBgn0	1179.	1.775166	0.242	3.192	0.001	0.095	CG11	3380
031734	178447	655	814766	419755	410861	488625	147	3
FBgn0	37.31	2.502008	0.367	4.081	4.47E-	0.005	CG90	3381
031747	851468	569	962502	960965	05	177352	21	8

FBgn0	2063.	2.457048	0.307	4.743	2.10E-	0.000	CG60	3402
031918	622471	651	180707	294802	06	30974	55	7
FBgn0	62.78	2.089744	0.292	3.723	0.000	0.018	CG73	3409
031976	110609	306	660552	577706	196419	217533	67	4
FBgn0	639.0	3.138659	0.337	6.336	2.35E-	1.02E-	CG67	3462
032399	601013	109	521008	373316	10	07	85	0
FBgn0	281.0	3.046362	0.366	5.579	2.41E-	5.80E-	ChLD	3500
032598	145396	585	752466	683239	08	06	3	2
FBgn0	121.0	-	0.382	-	0.001	0.082	CG13	3525
032809	419833	2.23774438	089836	3.239406	197786	859025	078	1
				714				
FBgn0	598.3	-	0.179	-	1.69E-	0.002	CG13	3525
032810	81953	1.773314306	722862	4.302815	05	132887	077	2
				432				
FBgn0	685.3	2.472291	0.306	4.797	1.60E-	0.000	CG12	3577
033252	860171	582	857151	970577	06	247763	769	0
FBgn0	14.52	2.604084	0.471	3.403	0.000	0.049	CG14	3580
033277	04193	773	34139	233427	665934	338193	760	2
FBgn0	3705.	2.715607	0.462	3.709	0.000	0.018	CG82	3590
033359	520771	514	473712	632506	20756	774509	13	2
FBgn0	354.7	2.810657	0.303	5.965	2.44E-	7.63E-	CG81	3590
033362	217517	024	51295	666457	09	07	72	5
FBgn0	2166.	2.979695	0.297	6.648	2.96E-	1.77E-	CG81	3590
033365	036299	243	766931	472476	11	08	70	8
FBgn0	1198.	3.293991	0.302	7.577	3.53E-	5.52E-	CG13	3618
033591	586745	815	7424	372088	14	11	216	1
FBgn0	206.2	2.440257	0.305	4.719	2.37E-	0.000	CG13	3625
033645	280837	68	179113	384841	06	344413	196	0
FBgn0	3270.	1.848011	0.206	4.112	3.91E-	0.004	Cpr4	3634
033725	447136	407	181781	930851	05	614667	9Ac	8
FBgn0	8291.	1.539726	0.153	3.509	0.000	0.036	CG13	3643
033789	694862	732	808741	077115	449665	324189	324	4
FBgn0	319.9	1.654431	0.177	3.688	0.000	0.019	GLaz	3644
033799	401669	673	406919	873452	225249	753158		7
FBgn0	1785.	2.742416	0.312	5.573	2.50E-	5.90E-	CG63	3653
033873	958312	28	633891	344184	08	06	37	0
FBgn0	864.3	2.582984	0.464	3.410	0.000	0.048	CG63	3653
033874	875756	795	203519	109425	649368	687061	47	1
FBgn0	3723.	2.485664	0.349	4.247	2.16E-	0.002	Cpr5	3661
033942	476803	048	792079	27756	05	630429	1A	3
FBgn0	3422.	3.095653	0.339	6.174	6.63E-	2.60E-	CG12	3671
034022	058761	683	398137	617527	10	07	964	4
FBgn0	13429	1.643877	0.198	3.243	0.001	0.082	CG10	3705
034295	.84478	972	528772	247644	181754	204153	911	8
FBgn0	1863.	2.496183	0.454	3.291	0.000	0.070	CG57	3706
034301	70151	6	553381	54652	996381	48413	56	4
FBgn0	3126.	2.920209	0.326	5.884	4.00E-	1.19E-	CG15	3716
034391	528667	936	32483	351299	09	06	080	7

FBgn0	3511.	-	0.266	-	2.09E-	8.71E-	CG10	3722
034440	464161	3.193595625	957392	8.217025	16	13	073	5
				215				
FBgn0	2316.	-	0.338	-	3.66E-	0.000	CG10	3722
034441	093438	2.565279811	098322	4.629658	06	515287	081	6
				617				
FBgn0	201.5	2.501854	0.381	3.936	8.27E-	0.008	CG15	3737
034563	651125	098	514718	556114	05	624809	649	1
FBgn0	150.9	3.035617	0.385	5.273	1.33E-	2.53E-	CG15	3737
034565	980751	131	973516	981363	07	05	650	3
FBgn0	1646.	2.593161	0.329	4.840	1.30E-	0.000	CG43	3748
034661	309969	43	154383	164712	06	205616	86	6
FBgn0	3264.	-	0.191	-	1.77E-	2.21E-	CG32	3754
034712	174879	3.15775266	393875	11.27388	29	25	64	0
				562				
FBgn0	503.0	3.338272	0.347	6.725	1.75E-	1.10E-	St1	3774
034887	059627	661	692259	121418	11	08		2
FBgn0	271.9	2.475585	0.272	5.410	6.28E-	1.33E-	CG43	3783
034956	316357	443	721252	599399	08	05	24	0
FBgn0	809.4	2.199595	0.308	3.883	0.000	0.010	CG14	3840
035428	703383	748	910323	313895	103042	239635	960	3
FBgn0	1452.	2.867468	0.318	5.854	4.78E-	1.39E-	CG12	3840
035429	571677	348	973924	611335	09	06	017	4
FBgn0	173.5	3.342023	0.385	6.068	1.29E-	4.61E-	CG15	3850
035508	583366	767	902486	952274	09	07	005	6
FBgn0	640.6	2.677183	0.326	5.132	2.86E-	5.12E-	CG11	3856
035557	155625	162	793681	238654	07	05	353	0
FBgn0	166.3	2.295881	0.233	5.547	2.90E-	6.66E-	lin-	3863
035626	649886	86	591391	64391	08	06	28	9
FBgn0	851.4	2.576808	0.334	4.707	2.51E-	0.000	CG13	3890
035845	435201	616	986299	083909	06	361649	675	7
FBgn0	98.26	2.260484	0.344	3.653	0.000	0.022	CG13	3902
035941	495142	359	973537	858122	258329	307159	313	2
FBgn0	840.8	2.714708	0.489	3.503	0.000	0.036	Cpr6	3922
036110	394539	131	491664	038476	459983	919534	7Fb	5
FBgn0	2161.	2.407517	0.311	4.515	6.31E-	0.000	CG12	3924
036131	780567	048	691442	738509	06	858729	522	8
FBgn0	792.5	1.582529	0.172	3.385	0.000	0.051	CG11	3931
036196	309707	324	058626	644411	710113	394967	658	9
FBgn0	571.4	1.491848	0.136	3.600	0.000	0.026	obst-	3935
036228	842534	302	591102	880986	317141	472794	G	5
FBgn0	2395.	2.338080	0.272	4.907	9.24E-	0.000	CG10	3942
036289	032659	567	675258	23132	07	150205	657	3
FBgn0	345.0	2.006802	0.250	4.021	5.78E-	0.006	CG13	3953
036382	961671	714	352256	544408	05	45829	737	1
FBgn0	433.1	-	0.238	-	1.15E-	2.22E-	CG51	3977
036575	188544	2.264070775	454263	5.301103	07	05	57	1
				698				

FBgn0	856.6	2.615435	0.475	3.396	0.000	0.049	CG73	4000
036780	358463	082	625762	441514	682681	987453	30	7
FBgn0	930.0	2.547079	0.324	4.762	1.91E-	0.000	CG13	4000
036781	477101	011	847083	483926	06	288472	699	8
FBgn0	28421	1.894607	0.241	3.708	0.000	0.018	CG72	4021
036948	.72702	789	226313	582942	208422	774509	98	0
FBgn0	162.2	2.202457	0.340	3.528	0.000	0.034	CG13	4021
036956	100151	279	813854	193666	418406	018575	813	8
FBgn0	72.46	2.634501	0.307	5.322	1.02E-	2.00E-	CG56	4024
036977	96503	254	102836	325503	07	05	65	3
FBgn0	4237.	2.137661	0.278	4.084	4.42E-	0.005	zye	4025
036985	03805	525	542264	340762	05	172475		5
FBgn0	590.1	2.293752	0.377	3.431	0.000	0.045	CG71	4039
037099	885778	504	029721	433738	6004	561247	73	2
FBgn0	15.78	2.691857	0.498	3.393	0.000	0.050	CG14	4048
037179	884717	667	607093	168069	690892	294552	453	4
FBgn0	92.52	2.559564	0.450	3.461	0.000	0.041	CG13	4050
037197	924304	662	534949	584202	537006	763065	239	3
FBgn0	3658.	3.070978	0.346	5.982	2.19E-	7.23E-	Twdl	4053
037225	104599	482	156995	772303	09	07	G	5
FBgn0	6331.	2.791184	0.351	5.090	3.57E-	6.29E-	Twdl	4053
037227	82759	742	859179	629571	07	05	V	7
FBgn0	1838.	2.114858	0.206	5.407	6.40E-	1.33E-	CG26	4064
037323	844261	506	174392	356839	08	05	63	9
FBgn0	206.3	3.047413	0.364	5.616	1.94E-	4.87E-	CG10	4074
037395	007536	927	511418	871862	08	06	280	0
FBgn0	685.0	3.060535	0.323	6.368	1.91E-	8.52E-	Osi2	4075
037409	994053	318	536395	789879	10	08	4	5
FBgn0	55.51	2.737382	0.448	3.874	0.000	0.010	Osi5	4075
037413	156657	059	471864	004586	107061	472779		9
FBgn0	97.53	3.777118	0.400	6.933	4.11E-	3.22E-	Osi1	4076
037417	603954	434	544607	356207	12	09	0	4
FBgn0	12688	3.007385	0.354	5.660	1.51E-	4.02E-	Osi1	4077
037424	.4681	399	618835	684655	08	06	5	1
FBgn0	655.5	3.276846	0.343	6.618	3.62E-	2.06E-	Osi1	4077
037427	234419	071	992374	885316	11	08	7	4
FBgn0	54.35	3.753032	0.394	6.979	2.97E-	2.66E-	CG14	4141
037940	356116	283	469413	07668	12	09	720	5
FBgn0	6949.	2.250778	0.338	3.699	0.000	0.019	CG41	4149
038017	029124	793	124572	165624	216309	345787	15	9
FBgn0	154.7	2.590173	0.221	7.173	7.30E-	9.13E-	CG67	4155
038070	350195	948	664508	786933	13	10	53	7
FBgn0	1006.	3.012169	0.304	6.599	4.11E-	2.24E-	CG15	4163
038132	855882	756	875579	970268	11	08	887	1
FBgn0	923.0	2.852268	0.334	5.539	3.03E-	6.66E-	CG14	4185
038315	509382	068	373273	521895	08	06	866	4
FBgn0	363.4	3.625249	0.337	7.789	6.75E-	2.11E-	CG10	4194
038394	609589	608	040236	128195	15	11	264	8

FBgn0	1221.	1.919287	0.237	3.874	0.000	0.010	CG89	4196
038405	174021	097	288258	136487	107003	472779	27	4
FBgn0	216.5	3.325792	0.403	5.763	8.25E-	2.30E-	Cad8	4200
038439	79382	785	55451	267974	09	06	9D	6
FBgn0	29.46	2.351366	0.361	3.736	0.000	0.017	CG14	4201
038447	901072	707	640204	771218	186398	548076	892	6
FBgn0	196.4	1.743468	0.186	3.980	6.86E-	0.007	CG52	4207
038485	795915	392	757312	933244	05	409542	55	3
FBgn0	2283.	2.999870	0.258	7.733	1.04E-	2.18E-	CG58	4210
038511	081735	086	589849	753239	14	11	73	0
FBgn0	423.0	2.504163	0.340	4.421	9.79E-	0.001	CG14	4211
038526	94618	588	168777	815557	06	263395	327	8
FBgn0	626.3	2.434477	0.273	5.243	1.57E-	2.89E-	CG17	4233
038717	982949	48	548165	966754	07	05	751	6
FBgn0	1425.	2.237778	0.320	3.864	0.000	0.010	CG74	4234
038727	956602	569	307215	348071	111386	811378	32	7
FBgn0	10269	1.837831	0.230	3.639	0.000	0.023	CG43	4241
038784	.70844	856	235034	028526	273668	310225	62	0
FBgn0	274.2	3.404024	0.365	6.578	4.74E-	2.47E-	CG13	4262
038958	3965	963	410484	970954	11	08	857	7
FBgn0	814.1	2.956147	0.311	6.280	3.37E-	1.36E-	CG13	4263
038967	957875	635	458565	603129	10	07	847	6
FBgn0	38.51	3.317311	0.398	5.812	6.15E-	1.75E-	CG70	4270
039027	726964	922	672532	56981	09	06	31	4
FBgn0	904.6	3.113628	0.302	6.992	2.70E-	2.60E-	CG17	4287
039167	29394	479	268113	561856	12	09	786	9
FBgn0	240.2	2.816051	0.367	4.941	7.76E-	0.000	CG13	4291
039200	44505	728	521628	346551	07	129525	616	7
FBgn0	985.8	2.699475	0.478	3.549	0.000	0.031	CG11	4299
039264	322974	409	855012	03962	386639	641214	786	8
FBgn0	35.14	2.962235	0.443	4.422	9.74E-	0.001	CG45	4308
039344	684974	641	663494	801667	06	263395	82	6
FBgn0	254.3	3.344159	0.385	6.085	1.16E-	4.27E-	MCO	4313
039387	21468	079	177816	914041	09	07	3	4
FBgn0	1933.	1.985694	0.265	3.712	0.000	0.018	CG56	4331
039527	073994	098	522106	286384	205395	774509	39	4
FBgn0	153.1	3.102357	0.348	6.025	1.69E-	5.72E-	CG14	4345
039648	104062	21	936981	034102	09	07	515	4
FBgn0	317.9	1.860571	0.263	3.263	0.001	0.077	CG15	4349
039686	784797	454	715841	252791	101412	043481	506	8
FBgn0	232.6	2.171471	0.209	5.584	2.35E-	5.76E-	CG97	4360
039758	510322	35	778225	332447	08	06	37	0
FBgn0	850.9	1.823793	0.210	3.910	9.21E-	0.009	yello	4377
039896	113595	637	656216	606834	05	37187	w-h	9
FBgn0	799.3	3.131009	0.288	7.395	1.41E-	1.96E-	CG11	3106
040359	923516	37	15691	308939	13	10	380	5
FBgn0	249.9	2.823746	0.261	6.963	3.33E-	2.78E-	CG14	3106
040360	676448	301	918116	039937	12	09	626	4

FBgn0	199.4	3.962775	0.386	7.665	1.78E-	3.18E-	CG15	5021
040743	056257	62	497679	700938	14	11	919	6
FBgn0	1477.	-	0.336	-	0.000	0.030	CG32	4040
043783	466614	2.197578101	252887	3.561539	368686	370506	444	6
			978					
FBgn0	177.1	2.312558	0.323	4.060	4.89E-	0.005	CG81	4113
043791	648172	546	2346	699399	05	518933	47	5
FBgn0	2470.	2.707029	0.262	6.496	8.20E-	3.95E-	CG30	3798
043792	115964	213	748294	823213	11	08	427	6
FBgn0	94.16	2.928785	0.390	4.935	8.00E-	0.000	Ilp3	3915
044050	830224	153	799489	485354	07	13172		1
FBgn0	437.2	1.626553	0.167	3.738	0.000	0.017	CG30	2466
050413	95572	735	616219	025706	185471	548076	413	01
FBgn0	384.7	2.628846	0.294	5.539	3.03E-	6.66E-	CG30	2466
050471	264778	554	025428	81526	08	06	471	33
FBgn0	1595.	2.496467	0.298	5.016	5.28E-	9.05E-	CG31	3185
051041	279987	019	33739	022351	07	05	041	67
FBgn0	1170.	2.224463	0.189	6.445	1.15E-	5.33E-	CG31	4279
051148	48899	952	960999	870264	10	08	148	6
FBgn0	2384.	2.649256	0.273	6.040	1.54E-	5.35E-	CG31	3186
051268	16131	576	04164	311555	09	07	268	52
FBgn0	1807.	-	0.131	-	8.39E-	0.008	CG31	3186
051288	037009	1.518578192	855429	3.932930	05	683638	288	64
			138					
FBgn0	532.2	1.696174	0.189	3.681	0.000	0.020	CG31	4230
051475	568151	663	100096	514072	231853	159939	475	3
FBgn0	97.02	2.521370	0.446	3.403	0.000	0.049	Osi1	3188
051561	923901	788	990151	589059	665067	338193	6	01
FBgn0	526.3	1.964862	0.214	4.507	6.57E-	0.000	CG31	3446
051871	401686	602	078808	043969	06	885046	871	3
FBgn0	2105.	2.390712	0.324	4.291	1.77E-	0.002	Cpr3	3189
051876	087876	001	061209	510248	05	200002	0F	97
FBgn0	80.70	3.007754	0.339	5.914	3.32E-	1.02E-	CG32	3910
052036	230495	784	44977	733074	09	06	036	5
FBgn0	276.3	1.940926	0.295	3.179	0.001	0.099	CG32	3261
052037	550629	864	891003	977952	472863	148991	037	83
FBgn0	3180.	2.790142	0.519	3.444	0.000	0.043	CG32	3179
052188	667606	036	666	793459	571496	632318	188	01
FBgn0	933.1	2.119493	0.225	4.965	6.84E-	0.000	Drsl2	3840
052279	522457	514	437035	881109	07	115717		8
FBgn0	26.21	2.692885	0.434	3.899	9.65E-	0.009	Twdl	3180
052569	81452	98	166251	165301	05	746676	Z	92
FBgn0	1692.	2.776798	0.320	5.545	2.93E-	6.66E-	CG32	3226
052645	411499	419	407271	437264	08	06	645	4
FBgn0	612.2	2.716429	0.491	3.495	0.000	0.037	CG33	3188
053003	696276	681	068694	294451	473539	765492	003	26
FBgn0	15.55	2.657231	0.499	3.315	0.000	0.065	CG33	2768
053257	724931	043	83789	537046	914672	071649	257	960

FBgn0	46.54	2.538743	0.399	3.854	0.000	0.011	CG33	2768
053341	943518	177	221064	363699	116031	175571	341	683
FBgn0	6778.	2.749856	0.469	3.727	0.000	0.018	dyl	3853
066365	379785	66	456715	407886	193459	076888		1
FBgn0	270.9	2.381500	0.374	3.691	0.000	0.019	CG34	4379
083951	500486	928	226929	612814	222837	753158	115	880
FBgn0	1811.	2.228169	0.339	3.618	0.000	0.025	CG17	4379
083978	219604	638	411943	522161	29629	039146	672	911
FBgn0	28.20	3.446473	0.446	5.479	4.27E-	9.21E-	CG34	5740
085250	301248	732	476939	507496	08	06	221	574
FBgn0	1297.	-	0.358	-	6.41E-	0.006	CG34	5740
085307	804474	2.434512489	8914	3.997065	05	982637	278	656
				65				
FBgn0	1038.	-	0.342	-	0.000	0.039	CG34	2768
085359	844324	2.191178871	471958	3.478179	504832	754738	330	869
				283				
FBgn0	333.6	2.821328	0.267	6.812	9.57E-	6.66E-	CG34	5740
085411	603528	435	339572	790266	12	09	382	788
FBgn0	54.61	2.734003	0.483	3.588	0.000	0.027	CG34	5740
085473	45945	252	167949	820936	332177	544284	444	745
FBgn0	1030.	1.759463	0.193	3.927	8.57E-	0.008	Inva	4958
086359	367418	861	3514	894305	05	794756	dolysin	0
FBgn0	1144.	3.172317	0.364	5.966	2.42E-	7.63E-	CG13	3501
086673	405303	977	089204	444361	09	07	272	3
FBgn0	422.6	3.476190	0.358	6.916	4.64E-	3.42E-	CG34	5740
250833	595185	951	023624	278106	12	09	461	547
FBgn0	1123.	2.204801	0.347	3.468	0.000	0.040	CG20	4061
250839	171955	634	325865	793297	522802	912487	16	2
FBgn0	629.1	2.178523	0.275	4.282	1.85E-	0.002	CG42	3226
250862	878903	292	183035	688762	05	266649	237	1
FBgn0	328.6	2.212757	0.255	4.751	2.02E-	0.000	CG42	3180
259167	910856	444	235632	520909	06	30094	272	20
FBgn0	449.3	2.853011	0.365	5.070	3.97E-	6.90E-	CG42	5354
259192	045321	055	452138	461658	07	05	296	4
FBgn0	159.4	3.012918	0.318	6.311	2.77E-	1.16E-	CG42	3331
259229	715749	962	947835	122835	10	07	329	2
FBgn0	2011.	2.767968	0.488	3.617	0.000	0.025	CG42	4294
259233	419033	658	785841	061927	297966	039146	331	8
FBgn0	640.4	1.639999	0.161	3.973	7.08E-	0.007	CG42	4255
259237	62088	857	057215	742226	05	571688	335	8
FBgn0	145.1	2.828067	0.323	5.652	1.58E-	4.04E-	CG42	8673
259748	888565	112	395097	736013	08	06	397	976
FBgn0	1091.	2.124334	0.250	4.490	7.09E-	0.000	mtg	4097
260386	148792	085	35638	934422	06	944555		0
FBgn0	422.6	2.363427	0.283	4.813	1.48E-	0.000	f	3271
262111	767546	222	242986	631007	06	231971		8
FBgn0	1268.	2.296528	0.243	5.335	9.54E-	1.90E-	CG43	1279
262811	222916	358	014172	196493	08	05	182	8293

FBgn0	101.8	4.100158	0.400	7.747	9.40E-	2.18E-	CG43	1279
262889	137617	818	166026	181453	15	11	244	8420
FBgn0	212.2	2.287214	0.324	3.967	7.28E-	0.007	CG43	1279
263020	563649	71	478817	022321	05	722293	315	8419
FBgn0	2962.	2.527443	0.465	3.279	0.001	0.073	CG43	3483
263038	390736	741	819683	045082	04159	268246	333	0
FBgn0	374.3	2.240283	0.319	3.884	0.000	0.010	CG43	1446
263760	951922	663	306618	303023	102624	239635	677	2616
FBgn0	1513.	2.068282	0.230	4.634	3.58E-	0.000	Hr4	3116
264562	428971	72	524771	13417	06	509995		2

References

- [1] B. Marte, Cell division and cancer, *Nature*. 432 (2004) 293. doi:10.1038/432293a.
- [2] P. Cheung, P. Lau, Epigenetic regulation by histone methylation and histone variants., *Mol. Endocrinol.* 19 (2005) 563–573. doi:10.1210/me.2004-0496.
- [3] T. Sorlie, C.M. Perou, R. Tibshirani, T. Aas, S. Geisler, H. Johnsen, et al., Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications., *Proc. Natl. Acad. Sci. U. S. A.* 98 (2001) 10869–74. doi:10.1073/pnas.191367098.
- [4] A. Ben-Dor, R. Shamir, Z. Yakhini, Clustering gene expression patterns., *J. Comput. Biol.* 6 (1999) 281–297. doi:10.1089/106652799318274.
- [5] M.B. Eisen, P.T. Spellman, P.O. Brown, D. Botstein, Cluster analysis and display of genome-wide expression patterns, *Proc. Natl. Acad. Sci.* 95 (1998) 14863–14868. doi:10.1073/pnas.95.25.14863.
- [6] G. Yi, S.-H. Sze, M.R. Thon, Identifying clusters of functionally related genes in genomes., *Bioinformatics.* 23 (2007) 1053–1060. doi:10.1093/bioinformatics/btl673.
- [7] M.H. Asyali, D. Colak, O. Demirkaya, M.S. Inan, Gene Expression Profile Classification: A Review, *Curr. Bioinform.* 1 (2006) 55–73. doi:10.2174/157489306775330615.
- [8] E. Blaveri, J.P. Simko, J.E. Korkola, J.L. Brewer, F. Baehner, K. Mehta, et al., Bladder cancer outcome and subtype classification by gene expression, *Clin. Cancer Res.* 11 (2005) 4044–4055. doi:10.1158/1078-0432.CCR-04-2409.
- [9] Z. Cai, R. Goebel, M.R. Salavatipour, G. Lin, Selecting dissimilar genes for multi-class classification, an application in cancer subtyping., *BMC Bioinformatics.* 8 (2007) 206. doi:10.1186/1471-2105-8-206.
- [10] R. Wesolowski, B. Ramaswamy, Gene expression profiling: Changing face of breast cancer classification and management, *Gene Expr.* 15 (2011) 105–115. doi:10.3727/105221611X13176664479241.
- [11] H. Hijazi, C. Chan, A classification framework applied to cancer gene expression profiles., *J. Healthc. Eng.* 4 (2013) 255–83. doi:10.1260/2040-2295.4.2.255.
- [12] a Antoniadis, S. Lambert-Lacroix, F. Leblanc, Effective dimension reduction methods for tumor classification using gene expression data, *Bioinformatics.* 19 (2003) 563–570. doi:10.1093/bioinformatics/btg062.
- [13] J. Cao, L. Zhang, B. Wang, F. Li, J. Yang, A fast gene selection method for multi-cancer classification using multiple support vector data description, *J. Biomed. Inform.* 53 (2015) 381–389. doi:10.1016/j.jbi.2014.12.009.
- [14] A. Jain, D. Zongker, Feature Selection: Evaluation, Application, and Small Sample Performance, *IEEE Trans. Pattern Anal. Mach. Intell.* 19 (1997) 153–158. doi:10.1109/34.574797.
- [15] C.E. Gillies, M.R. Siadat, N. V. Patel, G.D. Wilson, A simulation to analyze feature selection methods utilizing gene ontology for gene expression classification, *J. Biomed. Inform.* 46 (2013) 1044–1059. doi:10.1016/j.jbi.2013.07.008.
- [16] I. Guyon, J. Weston, S. Barnhill, V. Vapnik, Gene selection for cancer classification using support vector machines, *Mach. Learn.* 46 (2002) 389–422. doi:10.1023/A:1012487302797.
- [17] R.M. Luque-Baena, D. Urda, M. Gonzalo Claros, L. Franco, J.M. Jerez, Robust gene signatures from microarray data using genetic algorithms enriched with biological pathway keywords, *J. Biomed. Inform.* 49 (2014) 32–44. doi:10.1016/j.jbi.2014.01.006.
- [18] P.A. Mundra, J.C. Rajapakse, Gene and sample selection using T-score with sample selection., *J. Biomed. Inform.* 59 (2016) 31–41. doi:10.1016/j.jbi.2015.11.003.
- [19] Z. Mao, W. Cai, X. Shao, Selecting significant genes by randomization test for cancer classification using gene expression

- data, *J. Biomed. Inform.* 46 (2013) 594–601. doi:10.1016/j.jbi.2013.03.009.
- [20] E.R. Mardis, Next-generation DNA sequencing methods., *Annu. Rev. Genomics Hum. Genet.* 9 (2008) 387–402. doi:10.1146/annurev.genom.9.081307.164359.
- [21] J.C. Marioni, C.E. Mason, S.M. Mane, M. Stephens, Y. Gilad, RNA-seq: An assessment of technical reproducibility and comparison with gene expression arrays, *Genome Res.* 18 (2008) 1509–1517. doi:10.1101/gr.079558.108.
- [22] M. Bittner, P. Meltzer, Y. Chen, Y. Jiang, E. Seftor, M. Hendrix, et al., Molecular classification of cutaneous malignant melanoma by gene expression profiling., *Nature.* 406 (2000) 536–540. doi:10.1038/35020115.
- [23] L.J. Van't Veer, H. Dai, M.J. van De Vijver, Y.D. He, A.A. Hart, M. Mao, et al., Gene expression profiling predicts clinical outcome of breast cancer, *Nature.* 415 (2002) 530–536. doi:10.1038/415530a.
- [24] R. Rajabioun, Cuckoo optimization algorithm, *Appl. Soft Comput. J.* 11 (2011) 5508–5518. doi:10.1016/j.asoc.2011.05.008.
- [25] J.H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, *Q. Rev. Biol.* 1 (1975) 211. doi:10.1086/418447.
- [26] T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, et al., Molecular classification of cancer: class discovery and class prediction by gene expression monitoring., *Science.* 286 (1999) 531–537. doi:10.1126/science.286.5439.531.
- [27] B. Bioinformatics, T. Jirapech-Umpai, S. Aitken, Feature selection and classification for microarray data analysis: Evolutionary methods for identifying predictive genes, (n.d.). doi:10.1186/1471-2105-6-148.
- [28] a a Alizadeh, M.B. Eisen, R.E. Davis, C. Ma, I.S. Lossos, a Rosenwald, et al., Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling., *Nature.* 403 (2000) 503–11. doi:10.1038/35000501.
- [29] S. Dudoit, J. Fridlyand, T.P. Speed, Comparison of Discrimination Methods for the Classification of Tumors Using Gene Expression Data, *J. Am. Stat. Assoc.* 97 (2002) 77–87. doi:10.1198/016214502753479248.
- [30] D. Singh, P.G. Febbo, K. Ross, D.G. Jackson, J. Manola, C. Ladd, et al., Gene expression correlates of clinical prostate cancer behavior, *Cancer Cell.* 1 (2002) 203–209. doi:10.1016/S1535-6108(02)00030-2.
- [31] G.J. Tortora, B. Derrickson, *Principles of Anatomy and Physiology*, 2014. doi:10.1016/S0031-9406(05)60992-3.
- [32] DISEASES PICTURES, (2012). <http://diseasespictures.com/wp-content/uploads/2012/07/Cytoplasm-5.jpg> (accessed December 21, 2015).
- [33] A. Annunziato, DNA Packaging: Nucleosomes and Chromatin, *Nat. Educ.* 1 (2008) 26. <http://www.nature.com/scitable/topicpage/dna-packaging-nucleosomes-and-chromatin-310>.
- [34] B. Li, M. Carey, J.L. Workman, The Role of Chromatin during Transcription, *Cell.* 128 (2007) 707–719. doi:10.1016/j.cell.2007.01.015.
- [35] C. Mathé, M.-F. Sagot, T. Schiex, P. Rouzé, Current methods of gene prediction, their strengths and weaknesses., *Nucleic Acids Res.* 30 (2002) 4103–4117. doi:10.1093/nar/gkf543.
- [36] M.B. Avison, *Measuring Gene Expression*, New York, 2008. doi:10.1086/586945.
- [37] C.M. O'Connor, J.U. Adams, *Essentials of Cell Biology*, MA: NPG Education, Cambridge, 2010. <http://www.nature.com/scitable/ebooks/essentials-of-cell-biology-14749010>.
- [38] K. Kapur, Y. Xing, Z. Ouyang, W.H. Wong, Exon arrays provide accurate assessments of gene expression., *Genome Biol.* 8 (2007) R82. doi:10.1186/gb-2007-8-5-r82.
- [39] T.C. Mockler, J.R. Ecker, Applications of DNA tiling arrays for whole-genome analysis, *Genomics.* 85 (2005) 1–15. doi:10.1016/j.ygeno.2004.10.005.
- [40] J.B. Fan, K.L. Gunderson, M. Bibikova, J.M. Yeakley, J. Chen, E. Wickham Garcia, et al., [3] Illumina Universal Bead Arrays, *Methods Enzymol.* 410 (2006) 57–73. doi:10.1016/S0076-6879(06)10003-8.
- [41] C.A. Harrington, C. Rosenow, J. Retief, Monitoring gene expression using DNA microarrays, *Curr. Opin. Microbiol.* 3 (2000) 285–291. doi:10.1016/S1369-5274(00)00091-6.
- [42] G.D. Schuler, M.S. Boguski, E.A. Stewart, L.D. Stein, G. Gyapay, K. Rice, et al., A gene map of the human genome., *Science.* 274 (1996) 540–6. <http://www.ncbi.nlm.nih.gov/pubmed/8849440> (accessed March 21, 2016).
- [43] M.S. Boguski, T.M. Lowe, C.M. Tolstoshev, dbEST--database for "expressed sequence tags", *Nat. Genet.* 4 (1993) 332–3. doi:10.1038/ng0893-332.

- [44] D.A. Benson, M.S. Boguski, D.J. Lipman, J. Ostell, GenBank., *Nucleic Acids Res.* 25 (1997) 1–6. <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=146400&tool=pmcentrez&rendertype=abstract> (accessed March 21, 2016).
- [45] H. Ledford, The death of microarrays?, *Nature*. 455 (2008) 847–847. doi:10.1038/455847a.
- [46] D.B. Allison, X. Cui, G.P. Page, M. Sabripour, Microarray data analysis: from disarray to consolidation and consensus, *Nat Rev Genet.* 7 (2006) 55–65. doi:10.1038/nrg1749.
- [47] V. Trevino, F. Falciani, H.A. Barrera-Saldaña, DNA microarrays: a powerful genomic tool for biomedical and clinical research., *Mol. Med.* 13 (2007) 527–541. doi:10.2119/2006-00107.Trevino.
- [48] N. Jiang, L.J. Leach, X. Hu, E. Potokina, T. Jia, A. Druka, et al., Methods for evaluating gene expression from Affymetrix microarray datasets, *BMC Bioinformatics.* 9 (2008) 284. doi:10.1186/1471-2105-9-284.
- [49] M.D. Robinson, T.P. Speed, A comparison of Affymetrix gene expression arrays., *BMC Bioinformatics.* 8 (2007) 449. doi:10.1186/1471-2105-8-449.
- [50] M.B. Miller, Y.-W. Tang, Basic concepts of microarrays and potential applications in clinical microbiology., *Clin. Microbiol. Rev.* 22 (2009) 611–33. doi:10.1128/CMR.00019-09.
- [51] W.H. Koch, Technology platforms for pharmacogenomic diagnostic assays., *Nat. Rev. Drug Discov.* 3 (2004) 749–761. doi:10.1038/nrd1496.
- [52] P. Baldi, G.W. Hatfield, DNA microarrays and gene expression: from experiments to data analysis and modeling, Cambridge University Press, Cambridge, 2002.
- [53] NHGRI, DNA Microarray Technology, Natl. Hum. Genome Res. Inst. (2015). <https://www.genome.gov/10000533/dna-microarray-technology/> (accessed March 16, 2017).
- [54] F. Sanger, A.R. Coulson, A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase, *J. Mol. Biol.* 94 (1975). doi:10.1016/0022-2836(75)90213-2.
- [55] F. Sanger, S. Nicklen, a R. Coulson, DNA sequencing with chain-terminating inhibitors., *Proc. Natl. Acad. Sci. U. S. A.* 74 (1977) 5463–7. doi:10.1073/pnas.74.12.5463.
- [56] L.M. Smith, J.Z. Sanders, R.J. Kaiser, P. Hughes, C. Dodd, C.R. Connell, et al., Fluorescence detection in automated DNA sequence analysis., *Nature*. 321 (1986) 674–679. doi:10.1038/321674a0.
- [57] J. Shendure, H. Ji, Next-generation DNA sequencing, *Nat Biotechnol.* 26 (2008) 1135–1145. doi:10.1038/nbt1486.
- [58] M.L. Metzker, Sequencing technologies - the next generation., *Nat. Rev. Genet.* 11 (2010) 31–46. doi:10.1038/nrg2626.
- [59] C.S. Pareek, R. Smoczynski, A. Tretyn, Sequencing technologies and genome sequencing, *J. Appl. Genet.* 52 (2011) 413–435. doi:10.1007/s13353-011-0057-x.
- [60] O. Morozova, M.A. Marra, Applications of next-generation sequencing technologies in functional genomics, *Genomics.* 92 (2008) 255–264. doi:10.1016/j.ygeno.2008.07.001.
- [61] How is genome sequencing done?, 454 Life Sci. (2016). http://www.454.com/downloads/news-events/how-genome-sequencing-is-done_FINAL.pdf (accessed February 12, 2016).
- [62] European Bioinformatics Institute, 454 sequencing | EMBL-EBI Train online, EBI Online Train. Course. (2012). <https://www.ebi.ac.uk/training/online/course/ebi-next-generation-sequencing-practical-course/what-next-generation-dna-sequencing/454-seque> (accessed March 3, 2017).
- [63] S. Linnarsson, Recent advances in DNA sequencing methods - general principles of sample preparation, *Exp. Cell Res.* 316 (2010) 1339–1343. doi:10.1016/j.yexcr.2010.02.036.
- [64] W.J. Ansorge, Next-generation DNA sequencing techniques, *N. Biotechnol.* 25 (2009) 195–203. doi:10.1016/j.nbt.2008.12.009.
- [65] E.H. Margulies, Next-Generation Sequencing Technologies, NHGRI Curr. Top. Genome Anal. . (2010) 1–37. https://www.genome.gov/pages/research/intramuralresearch/dircalendar/currenttopicsingenomeanalysis2010/ctga2010_lec05_color.pdf (accessed April 3, 2017).
- [66] A. Valouev, J. Ichikawa, T. Tonthat, J. Stuart, S. Ranade, H. Peckham, et al., A high-resolution, nucleosome position map of *C. elegans* reveals a lack of universal sequence-dictated positioning, *Genome Res.* 18 (2008) 1051–1063. doi:10.1101/gr.076463.108.
- [67] J. Shendure, G.J. Porreca, N.B. Reppas, X. Lin, J.P. McCutcheon, A.M. Rosenbaum, et al., Accurate multiplex polony

- sequencing of an evolved bacterial genome., *Science*. 309 (2005) 1728–1732. doi:10.1126/science.1117389.
- [68] P.J.A. Cock, C.J. Fields, N. Goto, M.L. Heuer, P.M. Rice, The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants, *Nucleic Acids Res.* 38 (2009) 1767–1771. doi:10.1093/nar/gkp1137.
- [69] W.R. Pearson, D.J. Lipman, Improved tools for biological sequence comparison., *Proc. Natl. Acad. Sci. U. S. A.* 85 (1988) 2444–2448. doi:10.1073/pnas.85.8.2444.
- [70] B. Ewing, L. Hillier, M. Wendl, P. Green, Base-calling of automated sequencer traces using Phred. I. Accuracy assessment, *Genome Res.* 8 (1998) 175–185. doi:10.1101/gr.8.3.175.
- [71] B. Ewing, P. Green, Base-calling of automated sequencer traces using phred. II. Error probabilities, *Genome Res.* 8 (1998) 186–194. doi:10.1101/gr.8.3.175.
- [72] S. Deorowicz, S. Grabowski, Compression of genomic sequences in FASTQ format., *Bioinformatics*. 27 (2011) 1–3. doi:10.1093/bioinformatics/btr014.
- [73] R.K. Patel, M. Jain, NGS QC toolkit: A toolkit for quality control of next generation sequencing data, *PLoS One*. 7 (2012). doi:10.1371/journal.pone.0030619.
- [74] L. Soreq, N. Salomonis, A. Guffanti, H. Bergman, Z. Israel, H. Soreq, Whole transcriptome RNA sequencing data from blood leukocytes derived from Parkinson's disease patients prior to and following deep brain stimulation treatment, *Genomics Data*. 3 (2015) 57–60. doi:10.1016/j.gdata.2014.11.009.
- [75] A. Mortazavi, B. Williams, K. McCue, L. Schaeffer, B. Wold, Mapping and quantifying mammalian transcriptomes by RNA-Seq., *Nat. Methods*. 5 (2008) 621–628. doi:10.1038/nmeth.1226.
- [76] J.H. Malone, B. Oliver, Microarrays, deep sequencing and the true measure of the transcriptome., *BMC Biol.* 9 (2011) 34. doi:10.1186/1741-7007-9-34.
- [77] I. Nookaew, M. Papini, N. Pornputtapong, G. Scalcinati, L. Fagerberg, M. Uhlén, et al., A comprehensive comparison of RNA-Seq-based transcriptome analysis from reads to differential gene expression and cross-comparison with microarrays: A case study in *Saccharomyces cerevisiae*, *Nucleic Acids Res.* 40 (2012) 10084–10097. doi:10.1093/nar/gks804.
- [78] J.R. Bradford, Y. Hey, T. Yates, Y. Li, S.D. Pepper, C.J. Miller, A comparison of massively parallel nucleotide sequencing with oligonucleotide microarrays for global transcription profiling., *BMC Genomics*. 11 (2010) 282. doi:10.1186/1471-2164-11-282.
- [79] N. Raghavachari, J. Barb, Y. Yang, P. Liu, K. Woodhouse, D. Levy, et al., A systematic comparison and evaluation of high density exon arrays and RNA-seq technology used to unravel the peripheral blood transcriptome of sickle cell disease., *BMC Med. Genomics*. 5 (2012) 28. doi:10.1186/1755-8794-5-28.
- [80] S. Zhao, W.P. Fung-Leung, A. Bittner, K. Ngo, X. Liu, Comparison of RNA-Seq and microarray in transcriptome profiling of activated T cells, *PLoS One*. 9 (2014). doi:10.1371/journal.pone.0078644.
- [81] S.B. Montgomery, M. Sammeth, M. Gutierrez-Arcelus, R.P. Lach, C. Ingle, J. Nisbett, et al., Transcriptome genetics using second generation sequencing in a Caucasian population., *Nature*. 464 (2010) 773–7. doi:10.1038/nature08903.
- [82] D. O'Neil, H. Glowatz, M. Schlumpberger, Ribosomal RNA depletion for efficient use of RNA-seq capacity, *Curr. Protoc. Mol. Biol.* (2013). doi:10.1002/0471142727.mb0419s103.
- [83] W. Zhao, X. He, K.A. Hoadley, J.S. Parker, D.N. Hayes, C.M. Perou, Comparison of RNA-Seq by poly (A) capture, ribosomal RNA depletion, and DNA microarray for expression profiling., *BMC Genomics*. 15 (2014) 419. doi:10.1186/1471-2164-15-419.
- [84] S. He, O. Wurtzel, K. Singh, J.L. Froula, S. Yilmaz, S.G. Tringe, et al., Validation of two ribosomal RNA removal methods for microbial metatranscriptomics., *Nat. Methods*. 7 (2010) 807–12. doi:10.1038/nmeth.1507.
- [85] S.E. V Linsen, E. de Wit, G. Janssens, S. Heater, L. Chapman, R.K. Parkin, et al., Limitations and possibilities of small RNA digital gene expression profiling, *Nat Meth.* 6 (2009) 474–476. <http://dx.doi.org/10.1038/nmeth0709-474>.
- [86] K.R. Kukurba, S.B. Montgomery, RNA Sequencing and Analysis., *Cold Spring Harb. Protoc.* 2015 (2015) 951–69. doi:10.1101/pdb.top084970.
- [87] L. Li, L. Cheng, G. Jiang, A. Zhou, Biostatistics Challenges and Strategies for Differential Transcriptome Analysis from Microarray to Deep Sequencing in Statistics, *Ann. Biometrics Biostat.* 2 (2015). <https://www.jscimedcentral.com/Biometrics/biometrics-2-1014.pdf> (accessed April 17, 2017).

- [88] S. Tabakhi, A. Najafi, R. Ranjbar, P. Moradi, Gene selection for microarray data classification using a novel ant colony optimization, *Neurocomputing*. 168 (2015) 1024–1036. doi:10.1016/j.neucom.2015.05.022.
- [89] F. Ozsolak, P.M. Milos, RNA sequencing: advances, challenges and opportunities., *Nat. Rev. Genet.* 12 (2011) 87–98. doi:10.1038/nrg2934.
- [90] L. López-Kleine, C. González-Prieto, Challenges Analyzing RNA-Seq Gene Expression Data, *Open J. Stat.* 6 (2016) 628–636. doi:10.4236/ojs.2016.64053.
- [91] M.K. Kerr, Design considerations for efficient and effective microarray studies, *Biometrics*. 59 (2003) 822–828.
- [92] Y.H. Yang, T. Speed, Design issues for cDNA microarray experiments., *Nat. Rev. Genet.* 3 (2002) 579–588. doi:10.1038/nrg863.
- [93] D.B. Allison, G.L. Gadbury, M. Heo, J.R. Fernández, C.-K. Lee, T. a. Prolla, et al., A mixture model approach for the analysis of microarray gene expression data, *Comput. Stat. Data Anal.* 39 (2002) 1–20. doi:10.1016/S0167-9473(01)00046-9.
- [94] P. Pavlidis, Q. Li, W.S. Noble, The effect of replication on gene expression microarray experiments, *Bioinformatics*. 19 (2003) 1620–1627. doi:10.1093/bioinformatics/btg227.
- [95] K. Dobbin, R. Simon, Sample size determination in microarray experiments for class comparison and prognostic classification, *Biostatistics*. 6 (2005) 27–38. doi:10.1093/biostatistics/kxh015.
- [96] Z. Wu, A review of statistical methods for preprocessing oligonucleotide microarrays., *Stat. Methods Med. Res.* 18 (2009) 533–41. doi:10.1177/0962280209351924.
- [97] K. Shakya, H.J. Ruskin, G. Kerr, M. Crane, J. Becker, Comparison of microarray preprocessing methods, in: *Adv. Exp. Med. Biol.*, 2010: pp. 139–147. doi:10.1007/978-1-4419-5913-3_16.
- [98] J. Quackenbush, D. Dembélé, P. Kastner, Microarray data normalization and transformation, *BMC Bioinformatics*. 15 (2002) 14. doi:10.1038/ng1032.
- [99] T. Park, S.-G. Yi, S.-H. Kang, S. Lee, Y.-S. Lee, R. Simon, Evaluation of normalization methods for microarray data., *BMC Bioinformatics*. 4 (2003) 33. doi:10.1186/1471-2105-4-33.
- [100] A.K. Jain, R.C. Dubes, *Algorithms for Clustering Data*.pdf, 1988. doi:10.1126/science.311.5762.765.
- [101] B. Pontes, R. Giráldez, J.S. Aguilar-Ruiz, Biclustering on expression data: A review, *J. Biomed. Inform.* 57 (2015) 163–180. doi:10.1016/j.jbi.2015.06.028.
- [102] J.M. Claverie, Computational methods for the identification of differential and coordinated gene expression, *Hum. Mol. Genet.* 8 (1999) 1821–1832. doi:10.1093/hmg/8.10.1821.
- [103] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, A.Y. Wu, An efficient k-means clustering algorithm: analysis and implementation, *IEEE Trans. Pattern Anal. Mach. Intell.* 24 (2002) 881–892. doi:10.1109/TPAMI.2002.1017616.
- [104] D. Dembélé, P. Kastner, Fuzzy C-means method for clustering microarray data, *Bioinformatics*. 19 (2003) 973–980. doi:10.1093/bioinformatics/btg119.
- [105] S.C. Johnson, Hierarchical clustering schemes, *Psychometrika*. 32 (1967) 241–254. doi:10.1007/BF02289588.
- [106] F.D. Gibbons, F.P. Roth, Judging the quality of gene expression-based clustering methods using gene annotation, *Genome Res.* 12 (2002) 1574–1581. doi:10.1101/gr.397002.
- [107] T. Kohonen, Self-organized formation of topologically correct feature maps, *Biol. Cybern.* 43 (1982) 59–69. doi:10.1007/BF00337288.
- [108] C.D. Klose, Self-organizing maps for geoscientific data analysis: geological interpretation of multidimensional geophysical data, *Comput. Geosci.* 10 (2006) 265–277. doi:10.1007/s10596-006-9022-x.
- [109] B. Abu-Jamous, R. Fa, A.K. Nandi, *Integrative Cluster Analysis in Bioinformatics*, John Wiley & Sons, Ltd, Chichester, UK, 2015. doi:10.1002/9781118906545.
- [110] J. Einbeck, L. Evers, K. Hinchliff, Data Compression and Regression Based on Local Principal Curves, in: Springer, Berlin, Heidelberg, 2009: pp. 701–712. doi:10.1007/978-3-642-01044-6_64.
- [111] V. Laparra, S. Jiménez, G. Camps-Valls, J. Malo, Nonlinearities and Adaptation of Color Vision from Sequential Principal Curves Analysis, *Neural Comput.* 24 (2012) 2751–2788. doi:10.1162/NECO_a_00342.
- [112] J. Einbeck, G. Tutz, L. Evers, Local principal curves, *Stat. Comput.* 15 (2005) 301–313. doi:10.1007/s11222-005-4073-

- 8.
- [113] U. Ozertem, D. Erdogmus, Locally Defined Principal Curves and Surfaces, *J. Mach. Learn. Res.* 12 (2011) 1249–1286. <http://www.jmlr.org/papers/volume12/ozertem11a/ozertem11a.pdf> (accessed April 23, 2017).
- [114] B. Abu-Jamous, R. Fa, D.J. Roberts, A.K. Nandi, A. Naranuntarat, L. Wodicka, et al., Paradigm of Tunable Clustering Using Binarization of Consensus Partition Matrices (Bi-CoPaM) for Gene Discovery, *PLoS One.* 8 (2013) e56432. doi:10.1371/journal.pone.0056432.
- [115] B. Abu-Jamous, R. Fa, D.J. Roberts, A.K. Nandi, Yeast gene CMR1/YDL156W is consistently co-expressed with genes participating in DNA-metabolic processes in a variety of stringent clustering experiments, *J. R. Soc. Interface.* 10 (2013). <http://rsif.royalsocietypublishing.org/content/10/81/20120990> (accessed April 23, 2017).
- [116] C. Liu, B. Abu-Jamous, E. Brattico, A. Nandi, Clustering consistency in neuroimaging data analysis, in: 2015 12th Int. Conf. Fuzzy Syst. Knowl. Discov., IEEE, 2015: pp. 1118–1122. doi:10.1109/FSKD.2015.7382099.
- [117] B. Abu-Jamous, R. Fa, D.J. Roberts, A.K. Nandi, Comprehensive analysis of multiple microarray datasets by binarization of consensus partition matrix, in: 2012 IEEE Int. Work. Mach. Learn. Signal Process., IEEE, 2012: pp. 1–6. doi:10.1109/MLSP.2012.6349787.
- [118] B. Abu-Jamous, R. Fa, D.J. Roberts, A.K. Nandi, UNCLES: method for the identification of genes differentially consistently co-expressed in a specific subset of datasets., *BMC Bioinformatics.* 16 (2015) 184. doi:10.1186/s12859-015-0614-0.
- [119] B. Abu-Jamous, R. Fa, A. Nandi, D. Roberts, Binarization of consensus partition matrix for ensemble clustering, in: Proc. ... Eur. Signal Process. Conf. (EUSIPCO), [IEEE], 2012. <http://ieeexplore.ieee.org/document/6333845/> (accessed May 6, 2017).
- [120] B. Abu-Jamous, R. Fa, D.J. Roberts, A.K. Nandi, M-N scatter plots technique for evaluating varying-size clusters and setting the parameters of Bi-CoPaM and Uncles methods, in: 2014 IEEE Int. Conf. Acoust. Speech Signal Process., IEEE, 2014: pp. 6726–6730. doi:10.1109/ICASSP.2014.6854902.
- [121] N.S. Altman, An introduction to kernel and nearest-neighbor nonparametric regression, *Am. Stat.* 46 (1992) 175–185. doi:10.1080/00031305.1992.10475879.
- [122] T.S. Furey, N. Cristianini, N. Duffy, D.W. Bednarski, M. Schummer, D. Haussler, Support vector machine classification and validation of cancer tissue samples using microarray expression data, *Bioinformatics.* 16 (2000) 906–914. doi:10.1093/bioinformatics/16.10.906.
- [123] Z. Wang, Y. Wang, J. Xuan, Y. Dong, M. Bakay, Y. Feng, et al., Optimized multilayer perceptrons for molecular classification and diagnosis using genomic data., *Bioinformatics.* 22 (2006) 755–761. doi:10.1093/bioinformatics/btk036.
- [124] A.K. Jain, R.P.W. Duin, J. Mao, Statistical pattern recognition: a review, *IEEE Trans. Pattern Anal. Mach. Intell.* 22 (2000) 4–37. doi:10.1109/34.824819.
- [125] A. Statnikov, C.F. Aliferis, I. Tsamardinos, D. Hardin, S. Levy, A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis, *Bioinformatics.* 21 (2005) 631–643. doi:10.1093/bioinformatics/bti033.
- [126] C.J.C.J.C.J.C. Burges, A tutorial on support vector machines for pattern recognition, *Data Min. Knowl. Discov.* 2 (1998) 121–167. doi:10.1023/A:1009715923555.
- [127] U. Maulik, Analysis of gene microarray data in a soft computing framework, *Appl. Soft Comput. J.* 11 (2011) 4152–4160. doi:10.1016/j.asoc.2011.03.004.
- [128] H. Xiong, S. Szedmak, J. Piater, Scalable, accurate image annotation with joint SVMs and output kernels, *Neurocomputing.* 169 (2015) 205–214. doi:10.1016/j.neucom.2014.11.096.
- [129] S. Yin, J. Yin, Tuning kernel parameters for SVM based on expected square distance ratio, *Inf. Sci. (Ny).* 370 (2016) 92–102. doi:10.1016/j.ins.2016.07.047.
- [130] N. Cristianini, J. Shawe-Taylor, An introduction to support Vector Machines: and other kernel-based learning methods, (1999). <http://dl.acm.org/citation.cfm?id=345662> (accessed May 29, 2016).
- [131] S. Haykin, Neural networks-A comprehensive foundation, New York IEEE Press. Herrmann, M., Bauer, H.-U., Der, R. psychology (1994) pp107-116. doi:10.1017/S0269888998214044.
- [132] W.S. McCulloch, W.H. Pitts, A logical calculus of ideas imminent in nervous activity, *Bull. Math. Biophys.* 5 (1943) 115–133. doi:10.1007/BF02478259.

- [133] K.J. Hunt, D. Sbarbaro, R. Zbikowski, P.J. Gawthrop, Neural networks for control systems-A survey, *Automatica*. 28 (1992) 1083–1112. doi:10.1016/0005-1098(92)90053-I.
- [134] S.S. Haykin, *Neural Networks and Learning Machines*, 3rd ed., Pearson, New Jersey, 2009. <https://books.google.co.uk/books?id=KCwWOAAACAAJ>.
- [135] S. Osowski, K. Siwek, T. Markiewicz, MLP and SVM networks—a comparative study, *Proc. 6th Nord. Signal Process. Symp.* 2004 (2004) 37–40. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.76.6753&rep=rep1&type=pdf>.
- [136] A. Blumer, A. Ehrenfeucht, D. Haussler, M.K. Warmuth, Learnability and the Vapnik-Chervonenkis dimension, *J. ACM*. 36 (1989) 929–965. doi:10.1145/76359.76371.
- [137] I.S. Isa, Z. Saad, S. Omar, M.K. Osman, K.A. Ahmad, H.A.M. Sakim, Suitable MLP network activation functions for breast cancer and thyroid disease detection, in: *Proc. - 2nd Int. Conf. Comput. Intell. Model. Simulation, CIMSIm 2010*, 2010: pp. 39–44. doi:10.1109/CIMSIm.2010.93.
- [138] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, *Nature*. 323 (1986) 533–536. doi:10.1038/323533a0.
- [139] L. Bottou, Stochastic Gradient Learning in Neural Networks, *Proc. Neuro-Nimes*. 91 (1991).
- [140] M.J.D. Powell, Restart procedures for the conjugate gradient method, *Math. Program.* 12 (1977) 241–254. doi:10.1007/BF01593790.
- [141] F.D. Foresee, M.T. Hagan, Gauss-Newton approximation to Bayesian regularization, *Proc. 1997 Int. Jt. Conf. Neural Networks*. (1997) 1930–1935. doi:10.1109/ICNN.1997.614194.
- [142] M. Riedmiller, H. Braun, A direct adaptive method for faster backpropagation learning: The RPROP algorithm, in: *IEEE Int. Conf. Neural Networks - Conf. Proc.*, 1993: pp. 586–591. doi:10.1109/ICNN.1993.298623.
- [143] M.F. Moller, A Scaled Conjugate-Gradient Algorithm for Fast Supervised Learning, *Neural Networks*. 6 (1993) 525–533. doi:10.1016/S0893-6080(05)80056-5.
- [144] M.T. Hagan, M.B. Menhaj, Training Feedforward Networks with the Marquardt Algorithm, *IEEE Trans. Neural Networks*. 5 (1994) 989–993. doi:10.1109/72.329697.
- [145] T.R. Shultz, S.E. Fahlman, S. Craw, P. Andritsos, P. Tsaparas, R. Silva, et al., Curse of Dimensionality, in: *Encycl. Mach. Learn.*, Springer US, Boston, MA, 2011: pp. 257–258. doi:10.1007/978-0-387-30164-8_192.
- [146] Y. Saeys, I. Inza, P. Larrañaga, A review of feature selection techniques in bioinformatics, *Bioinformatics*. 23 (2007) 2507–2517. doi:10.1093/bioinformatics/btm344.
- [147] A.L. Blum, P. Langley, Selection of relevant features and examples in machine learning, *Artif. Intell.* 97 (1997) 245–271. doi:10.1016/S0004-3702(97)00063-5.
- [148] Y.Q. Zhang, J.C. Rajapakse, *Machine Learning in Bioinformatics*, John Wiley & Sons, Inc, New Jersey, 2008. doi:10.1002/9780470397428.
- [149] I. Inza, B. Sierra, R. Blanco, Gene selection by sequential search wrapper approaches in microarray cancer class prediction, *J. Intell. Fuzzy Syst.* 12 (2002) 25–33.
- [150] I. Inza, P. Larrañaga, R. Blanco, A.J. Cerrolaza, Filter versus wrapper gene selection approaches in DNA microarray domains, *Artif. Intell. Med.* 31 (2004) 91–103. doi:10.1016/j.artmed.2004.01.007.
- [151] R. Ruiz, J.C. Riquelme, J.S. Aguilar-Ruiz, Incremental wrapper-based gene selection from microarray data for cancer classification, *Pattern Recognit.* 39 (2006) 2383–2392. doi:10.1016/j.patcog.2005.11.001.
- [152] I. Guyon, S. Gunn, M. Nikravesh, L.A. Zadeh, eds., *Feature Extraction: Foundations and Applications*, Springer, Berlin, 2006.
- [153] V. Bolón-Canedo, N. Sánchez-Marroño, A. Alonso-Betanzos, A review of feature selection methods on synthetic data, *Knowl. Inf. Syst.* 34 (2012) 483–519. doi:10.1007/s10115-012-0487-8.
- [154] S. Li, X. Wu, M. Tan, Gene selection using hybrid particle swarm optimization and genetic algorithm, *Soft Comput.* 12 (2008) 1039–1048. doi:10.1007/s00500-007-0272-x.
- [155] B. Sahu, D. Mishra, A novel feature selection algorithm using particle swarm optimization for cancer microarray data, in: *Procedia Eng.*, 2012: pp. 27–31. doi:10.1016/j.proeng.2012.06.005.
- [156] G. Chandrashekar, F. Sahin, A survey on feature selection methods, *Comput. Electr. Eng.* 40 (2014) 16–28.

doi:10.1016/j.compeleceng.2013.11.024.

- [157] J.R. Cano, F. Herrera, M. Lozano, Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study, *IEEE Trans. Evol. Comput.* 7 (2003) 561–575. doi:10.1109/TEVC.2003.819265.
- [158] C. Garcia-Orsorio, A. de Haro-Garcia, N. Garcia-Pedrajas, Democratic instance selection: A linear complexity instance selection algorithm based on classifier ensemble concepts, *Artif. Intell.* 174 (2010) 410–441. doi:10.1016/j.artint.2010.01.001.
- [159] K.A. Ross, C.S. Jensen, R. Snodgrass, C.E. Dyreson, C.S. Jensen, R. Snodgrass, et al., Cross-Validation, in: *Encycl. Database Syst.*, Springer US, Boston, MA, 2009: pp. 532–538. doi:10.1007/978-0-387-39940-9_565.
- [160] H.M. Nguyen, I. Couckuyt, L. Knockaert, T. Dhaene, D. Gorissen, Y. Saeys, An alternative approach to avoid overfitting for surrogate models, in: *Proc. 2011 Winter Simul. Conf.*, IEEE, 2011: pp. 2760–2771. doi:10.1109/WSC.2011.6147981.
- [161] Yi Liu, T. Khoshgoftaar, Reducing overfitting in genetic programming models for software quality classification, in: *Eighth IEEE Int. Symp. High Assur. Syst. Eng. 2004. Proceedings.*, IEEE, n.d.: pp. 56–65. doi:10.1109/HASE.2004.1281730.
- [162] D.; Nee, Common Pitfalls in Machine Learning, (2015). <http://danielnee.com/2015/01/common-pitfalls-in-machine-learning/> (accessed May 21, 2017).
- [163] Y. Zhang, Y. Yang, Cross-validation for selecting a model selection procedure, *J. Econom.* 187 (2015) 95–112. doi:10.1016/j.jeconom.2015.02.006.
- [164] S. Yadav, S. Shukla, Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification, in: *2016 IEEE 6th Int. Conf. Adv. Comput.*, IEEE, 2016: pp. 78–83. doi:10.1109/IACC.2016.25.
- [165] A. Rahideh, M.H. Shaheed, Cancer classification using clustering based gene selection and artificial neural networks, in: *2nd Int. Conf. Control. Instrum. Autom.*, IEEE, 2011: pp. 1175–1180. doi:10.1109/ICCIAutom.2011.6356828.
- [166] J. Kennedy, R. Eberhart, Particle swarm optimization, *Proc. ICNN'95 - Int. Conf. Neural Networks.* 4 (1995) 1942–1948. doi:10.1109/ICNN.1995.488968.
- [167] R.L. Haupt, S.E. Haupt, *Practical Genetic Algorithms*, 2nd Editio, John Wiley & Sons, Inc, New Jersey, 2004.
- [168] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1st ed., Boston, MA, USA, 1989. doi:10.1007/s10589-009-9261-6.
- [169] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press Cambridge, Massachusetts, 1996.
- [170] J. Kennedy, R.C. Eberhart, Y. Shi, *Swarm Intelligence*, Morgan Kaufmann Publishers, San Francisco, 2001.
- [171] M. Clerc, J. Kennedy, The particle swarm - explosion, stability, and convergence in a multidimensional complex space, *IEEE Trans. Evol. Comput.* 6 (2002) 58–73. doi:10.1109/4235.985692.
- [172] I. Fister, X.-S. Yang, D. Fister, Cuckoo Search and Firefly Algorithm: Theory and Applications, in: X.-S. Yang (Ed.), *Springer International Publishing*, Cham, 2014: pp. 49–62. doi:10.1007/978-3-319-02141-6_3.
- [173] H. Kahramanli, A Modified Cuckoo Optimization Algorithm for Engineering Optimization, *Int. J. Futur. Comput. Commun.* 1 (2012) 199–201. doi:10.7763/IJFCC.2012.V1.52.
- [174] V. Elyasigomari, M.S. Mirjafari, H.R.C. Screen, M.H. Shaheed, Cancer classification using a novel gene selection approach by means of shuffling based on data clustering with optimization, *Appl. Soft Comput.* 35 (2015) 43–51. doi:10.1016/j.asoc.2015.06.015.
- [175] S. Wang, D. Li, X. Song, Y. Wei, H. Li, A feature selection method based on improved fisher's discriminant ratio for text sentiment classification, *Expert Syst. Appl.* 38 (2011) 8696–8702. doi:10.1016/j.eswa.2011.01.077.
- [176] M.-W. Mak, S.-Y. Kung, Fusion of feature selection methods for pairwise scoring SVM, *Neurocomputing.* 71 (2008) 3104–3113. doi:10.1016/j.neucom.2008.04.024.
- [177] V.N. Vapnik, An overview of statistical learning theory., *IEEE Trans. Neural Netw.* 10 (1999) 988–99. doi:10.1109/72.788640.
- [178] V. Cherkassky, The nature of statistical learning theory~, *IEEE Trans. Neural Netw.* 8 (1997) 1564. doi:10.1109/TNN.1997.641482.
- [179] Q. Shen, W.-M. Shi, W. Kong, B.-X. Ye, A combination of modified particle swarm optimization algorithm and support vector machine for gene selection and tumor classification., *Talanta.* 71 (2007) 1679–1683. doi:10.1016/j.talanta.2006.07.047.
- [180] K. Levenberg, A method for the solution of certain non-linear problems in least squares, *Q. Appl. Math.* 2 (1944) 164–

- [181] R. Cordeiro De Amorim, B. Mirkin, Minkowski metric, feature weighting and anomalous cluster initializing in K-Means clustering, *Pattern Recognit.* 45 (2012) 1061–1075. doi:10.1016/j.patcog.2011.08.012.
- [182] J. Macqueen, Some methods for classification and analysis of multivariate observations, *Proc. Fifth Berkeley Symp. Math. Stat. Probab.* 1 (1967) 281–297. doi:citeulike-article-id:6083430.
- [183] M.K. Gould, J. Fletcher, M.D. Iannettoni, W.R. Lynch, D.E. Midthun, D.P. Naidich, et al., Evaluation of patients with pulmonary nodules: When is it lung cancer? ACCP evidence-based clinical practice guidelines (2nd edition), *Chest*. 132 (2007). doi:10.1378/chest.07-1353.
- [184] B. Chandra, K. V. Naresh Babu, Classification of gene expression data using Spiking Wavelet Radial Basis Neural Network, *Expert Syst. Appl.* 41 (2014) 1326–1330. doi:10.1016/j.eswa.2013.08.030.
- [185] T. Nguyen, A. Khosravi, D. Creighton, S. Nahavandi, Hidden Markov models for cancer classification using gene expression profiles, *Inf. Sci. (Ny)*. 316 (2015) 293–307. doi:10.1016/j.ins.2015.04.012.
- [186] R.K. Sivagaminathan, S. Ramakrishnan, A hybrid approach for feature subset selection using neural networks and ant colony optimization, *Expert Syst. Appl.* 33 (2007) 49–60. doi:10.1016/j.eswa.2006.04.010.
- [187] J. Jaeger, R. Sengupta, W.L. Ruzzo, Improved gene selection for classification of microarrays., *Pac. Symp. Biocomput.* 64 (2003) 53–64.
- [188] S.-B. Cho, H.-H. Won, Machine learning in DNA microarray analysis for cancer classification, *Proc. First Asia-Pacific Bioinforma. Conf. Bioinforma.* 2003-Volume 19. (2003) 189–198.
- [189] C.P. Lee, W.S. Lin, Y.M. Chen, B.J. Kuo, Gene selection and sample classification on microarray data based on adaptive genetic algorithm/k-nearest neighbor method, *Expert Syst. Appl.* 38 (2011) 4661–4667. doi:10.1016/j.eswa.2010.07.053.
- [190] Y. Peng, W. Li, Y. Liu, A Hybrid Approach for Biomarker Discovery from Microarray Gene Expression Data for Cancer Classification, *Cancer Inform.* 2 (2006) 301–311.
- [191] Y. Liu, U. Aickelin, J. Feyereisl, L.G. Durrant, Wavelet feature extraction and genetic algorithm for biomarker detection in colorectal cancer data, *Knowledge-Based Syst.* 37 (2013) 502–514. doi:10.1016/j.knsys.2012.09.011.
- [192] O.H. Fang, N. Mustapha, M.N. Sulaiman, Integrative Gene Selection for Classification of Microarray Data, *Comput. Inf. Sci.* 4 (2011). www.ccsenet.org/cis (accessed September 3, 2016).
- [193] H. Peng, F. Long, C. Ding, Feature selection based on mutual information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy, *IEEE Trans. Pattern Anal. Mach. Intell.* 27 (2005) 1226–1238. doi:10.1109/TPAMI.2005.159.
- [194] M. Richeldi, M. Rossotto, Machine Learning: ECML-95: 8th European Conference on Machine Learning Heracleion, Crete, Greece, April 25--27, 1995 Proceedings, in: N. Lavrac, S. Wrobel (Eds.), Springer Berlin Heidelberg, Berlin, Heidelberg, 1995: pp. 335–338. doi:10.1007/3-540-59286-5_81.
- [195] B.S. Chlebus, S.H. Nguyen, Rough Sets and Current Trends in Computing: First International Conference, RSCTC'98 Warsaw, Poland, June 22--26, 1998 Proceedings, in: L. Polkowski, A. Skowron (Eds.), Springer Berlin Heidelberg, Berlin, Heidelberg, 1998: pp. 537–544. doi:10.1007/3-540-69115-4_74.
- [196] A. Tillander, Effect of data discretization on the classification accuracy in a high-dimensional framework, *Int. J. Intell. Syst.* 27 (2012) 355–374. doi:10.1002/int.21527.
- [197] M. Elloumi, A.Y. Zomaya, Biological Knowledge Discovery Handbook: Preprocessing, Mining and Postprocessing of Biological Data, (2013). <http://dl.acm.org/citation.cfm?id=2613615> (accessed May 17, 2016).
- [198] C.A. Gallo, R.L. Cecchini, J.A. Carballido, S. Micheletto, I. Ponzoni, Discretization of gene expression data revised, *Briefings Bioinforma.* . (2015). doi:10.1093/bib/bbv074.
- [199] W. Li, Y. Yang, How Many Genes Are Needed for a Discriminant Microarray Data Analysis?, *Methods Microarray Data Anal.* (2001) 8. <http://arxiv.org/abs/physics/0104029>.
- [200] T.M. Cover, J.A. Thomas, Entropy, relative entropy and mutual information, 1991. doi:10.1002/047174882X.ch2.
- [201] L. Li, W. Jiang, X. Li, K.L. Moser, Z. Guo, L. Du, et al., A robust hybrid between genetic algorithm and support vector machine for extracting an optimal feature gene subset, *Genomics*. 85 (2005) 16–23. doi:10.1016/j.ygeno.2004.09.007.
- [202] A.Y. Ng, Preventing Overfitting of Cross-Validation Data, in: *ICML '97 Proc. Fourteenth Int. Conf. Mach. Learn.*, 1997: pp. 245–253. doi:10.1007/s13398-014-0173-7.2.

- [203] Zong Woo Geem, Joong Hoon Kim, G.V. Loganathan, A New Heuristic Optimization Algorithm: Harmony Search, *Simulation*. 76 (2001) 60–68. doi:10.1177/003754970107600201.
- [204] Y. Wang, Y. Liu, L. Feng, X. Zhu, Novel feature selection method based on harmony search for email classification, *Knowledge-Based Syst.* 73 (2015) 311–323. doi:10.1016/j.knosys.2014.10.013.
- [205] Z.W. Geem, Novel derivative of harmony search algorithm for discrete design variables, *Appl. Math. Comput.* 199 (2008) 223–230. doi:10.1016/j.amc.2007.09.049.
- [206] E. Valian, S. Tavakoli, S. Mohanna, An intelligent global harmony search approach to continuous optimization problems, *Appl. Math. Comput.* 232 (2014) 670–684. doi:10.1016/j.amc.2014.01.086.
- [207] O.M. Alia, R. Mandava, D. Ramachandram, M.E. Aziz, Harmony search-based cluster initialization for fuzzy c-means segmentation of MR images, in: *TENCON 2009 - 2009 IEEE Reg. 10 Conf., IEEE, 2009*: pp. 1–6. doi:10.1109/TENCON.2009.5396049.
- [208] S.S. Shreem, S. Abdullah, M.Z.A. Nazri, Hybridising harmony search with a Markov blanket for gene selection problems, *Inf. Sci. (Ny)*. 258 (2014) 108–121. doi:10.1016/j.ins.2013.10.012.
- [209] M. Hadwan, M. Ayob, N.R. Sabar, R. Qu, A harmony search algorithm for nurse rostering problems, *Inf. Sci. (Ny)*. 233 (2013) 126–140. doi:10.1016/j.ins.2012.12.025.
- [210] J.A. Magee, T. Araki, S. Patil, T. Ehrig, L. True, P.A. Humphrey, et al., Expression profiling reveals hepsin overexpression in prostate cancer, *Cancer Res.* 61 (2001) 5692–5696.
- [211] O. Klezovitch, J. Chevillet, J. Mirosevich, R.L. Roberts, R.J. Matusik, V. Vasioukhin, Hepsin promotes prostate cancer progression and metastasis, *Cancer Cell*. 6 (2004) 185–195. doi:10.1016/j.ccr.2004.07.008.
- [212] W.M. Schmidt, M. Kalipciyan, E. Dornstauder, B. Rizovski, G.G. Steger, R. Sedivy, et al., Dissecting progressive stages of 5-fluorouracil resistance in vitro using RNA expression profiling, *Int. J. Cancer*. 112 (2004) 200–212. doi:10.1002/ijc.20401.
- [213] Y. Yap, X. Zhang, M. Ling, X. Wang, Y. Wong, A. Danchin, Classification between normal and tumor tissues based on the pair-wise gene expression ratio, *BMC Cancer*. 72 (2004). doi:10.1186/1471-2407-4-72.
- [214] E. Glaab, J. Bacardit, J.M. Garibaldi, N. Krasnogor, I. Plaza-Menacho, Using Rule-Based Machine Learning for Candidate Disease Gene Prioritization and Sample Classification of Cancer Gene Expression Data, *PLoS One*. 7 (2012) e39932. doi:10.1371/journal.pone.0039932.
- [215] E.J. van der Gaag, M.-T. Leccia, S.K. Dekker, N.L. Jalbert, D.M. Amodeo, H. Randolph Byers, Role of Zyxin in Differential Cell Spreading and Proliferation of Melanoma Cells and Melanocytes, *J. Invest. Dermatol.* 118 (2002) 246–254. doi:10.1046/j.0022-202x.2001.01657.x.
- [216] D. V. Nguyen, D.M. Rocke, Tumor classification by partial least squares using microarray gene expression data., *Bioinformatics*. 18 (2002) 39–50. <http://www.ncbi.nlm.nih.gov/pubmed/11836210> (accessed June 11, 2017).
- [217] Y. Wang, I. V. Tetko, M.A. Hall, E. Frank, A. Facius, K.F.X. Mayer, et al., Gene selection from microarray data for cancer classification—a machine learning approach, *Comput. Biol. Chem.* 29 (2005) 37–46. doi:10.1016/j.compbiolchem.2004.11.001.
- [218] J. Li, L. Wong, Identifying good diagnostic gene groups from gene expression profiles using the concept of emerging patterns., *Bioinformatics*. 18 (2002) 725–34. <http://www.ncbi.nlm.nih.gov/pubmed/12050069> (accessed June 11, 2017).
- [219] P. Zhou, N.B. Levy, H. Xie, L. Qian, C.-Y.G. Lee, R.D. Gascoyne, et al., MCL1 transgenic mice exhibit a high incidence of B-cell lymphoma manifested as a spectrum of histologic subtypes, *Blood*. 97 (2001) 3902–3909.
- [220] Y. Kambayashi, M. Mohania, W. Wöss, Data Warehousing and Knowledge Discovery: 6th International Conference, DaWaK 2004, Zaragoza, Spain, September 1-3, 2004. *Proceedings, Springer-Verlag Berlin Heidelberg, 2004*. https://books.google.co.uk/books?id=VdXzBwAAQBAJ&pg=PA284&lpg=PA284&dq=GENE3968X&source=bl&ots=5WOaVw_6ay&sig=sf2OLVxLDBKhSdFJ9ZIGbpPSobU&hl=en&sa=X&ved=0ahUKewi42_3-mLbUAhXFLsAKHSenAvAQ6AEIMTAD#v=onepage&q=GENE3968X&f=false (accessed June 11, 2017).
- [221] T. Matsuo, K. Kuriyama, Y. Miyazaki, S. Yoshida, M. Tomonaga, N. Emi, et al., The percentage of myeloperoxidase-positive blast cells is a strong independent prognostic factor in acute myeloid leukemia, even in the patients with normal karyotype, *Leukemia*. 17 (2003) 1538–1543. doi:10.1038/sj.leu.2403010.

- [222] R.F. Melhem, X.X. Zhu, N. Hailat, J.R. Strahler, S.M. Hanash, Characterization of the gene for a proliferation-related phosphoprotein (oncoprotein 18) expressed in high amounts in acute leukemia, *J. Biol. Chem.* 266 (1991) 17747–17753.
- [223] G. Roos, G. Brattsand, G. Landberg, Expression of Oncoprotein-18 in Human Leukemias and Lymphomas, 7 (1993).
- [224] a Kumatori, K. Tanaka, N. Inamura, S. Sone, T. Ogura, T. Matsumoto, et al., Abnormally high expression of proteasomes in human leukemic cells., *Proc. Natl. Acad. Sci. U. S. A.* 87 (1990) 7071–7075. doi:10.1073/pnas.87.18.7071.
- [225] H. Katoh, K. Hiramoto, M. Negishi, Activation of Rac1 by RhoG regulates cell migration., *J. Cell Sci.* 119 (2006) 56–65. doi:10.1242/jcs.02720.
- [226] D. Yamazaki, S. Kurisu, T. Takenawa, Regulation of cancer cell motility through actin reorganization, *Cancer Sci.* 96 (2005) 379–386. doi:10.1111/j.1349-7006.2005.00062.x.
- [227] Y. Li, B.F. Shen, C. Karanes, L. Sensenbrenner, B. Chen, Association between Lyn protein tyrosine kinase (p53/56lyn) and the beta subunit of the granulocyte-macrophage colony-stimulating factor (GM-CSF) receptors in a GM-CSF-dependent human megakaryocytic leukemia cell line (M-07e)., *J. Immunol.* 155 (1995) 2165–2174.
- [228] Y. Dai, M. Rahmani, S.J. Corey, P. Dent, S. Grant, A Bcr/Abl-independent, Lyn-dependent Form of Imatinib Mesylate (STI-571) Resistance Is Associated with Altered Expression of Bcl-2, *J. Biol. Chem.* . 279 (2004) 34227–34239. doi:10.1074/jbc.M402290200.
- [229] R. Tiedt, B.A. Bartholdy, G. Matthias, J.W. Newell, P. Matthias, The RING finger protein Siah-1 regulates the level of the transcriptional coactivator OBF-1, *EMBO J.* 20 (2001) 4143–4152. doi:10.1093/emboj/20.15.4143.
- [230] O.H. Krämer, R.H. Stauber, G. Bug, J. Hartkamp, S.K. Knauer, SIAH proteins: Critical roles in leukemogenesis, *Leukemia.* (2012) 792–802. doi:10.1038/leu.2012.284.
- [231] C. Heit, B.C. Jackson, M. McAndrews, M.W. Wright, D.C. Thompson, G. a Silverman, et al., Update of the human and mouse SERPIN gene superfamily., *Hum. Genomics.* 7 (2013) 22. doi:10.1186/1479-7364-7-22.
- [232] F. Yagasaki, D. Wakao, Y. Yokoyama, Y. Uchida, I. Murohashi, H. Kayano, et al., Fusion of ETV6 to fibroblast growth factor receptor 3 in peripheral T-cell lymphoma with a t(4;12)(p16;p13) chromosomal translocation, *Cancer Res.* 61 (2001) 8371–8374.
- [233] C.K. Mavis, S.R. Morey Kinney, B.A. Foster, A.R. Karpf, Expression level and DNA methylation status of glutathione-S-transferase genes in normal murine prostate and TRAMP tumors, *Prostate.* 69 (2009) 1312–1324. doi:10.1002/pros.20976.
- [234] J.-W. Liu, J.-J. Shen, A. Tanzillo-Swartz, B. Bhatia, C.M. Maldonado, M.D. Person, et al., Annexin II expression is reduced or lost in prostate cancer cells and its re-expression inhibits prostate cancer cell migration, *Oncogene.* 22 (2003) 1475–1485. doi:10.1038/sj.onc.1206196.
- [235] W. Xin, D.R. Rhodes, C. Ingold, A.M. Chinnaiyan, M.A. Rubin, Dysregulation of the annexin family protein family is associated with prostate cancer progression., *Am. J. Pathol.* 162 (2003) 255–61. doi:10.1016/S0002-9440(10)63816-3.
- [236] M. Igawa, D.B. Rukstalis, T. Tanabe, G.W. Chodak, High Levels of nm23 Expression Are Related to Cell Proliferation in Human Prostate Cancer, *Cancer Res.* 54 (1994) 1313–1318.
- [237] N. Konishi, S. Nakaoka, T. Tsuzuki, K. Matsumoto, Y. Kitahori, Y. Hiasa, et al., Expression of nm23-H1 and nm23-H2 proteins in prostate carcinoma., *Jpn. J. Cancer Res.* 84 (1993) 1050–4.
- [238] K. Imberg-Kazdan, S. Ha, A. Greenfield, C.S. Poultnery, R. Bonneau, S.K. Logan, et al., A genome-wide RNA interference screen identifies new regulators of androgen receptor function in prostate cancer cells, *Genome Res.* 23 (2013) 581–591. doi:10.1101/gr.144774.112.
- [239] D. Sims, I. Sudbery, N.E. Illott, A. Heger, C.P. Ponting, Sequencing depth and coverage: key considerations in genomic analyses., *Nat. Rev. Genet.* 15 (2014) 121–32. doi:10.1038/nrg3642.
- [240] D.A. Jaitin, E. Kenigsberg, H. Keren-Shaul, N. Elefant, F. Paul, I. Zaretsky, et al., Massively parallel single-cell RNA-seq for marker-free decomposition of tissues into cell types., *Science.* 343 (2014) 776–9. doi:10.1126/science.1247651.
- [241] M.K. Iyer, A.M. Chinnaiyan, C.A. Maher, ChimeraScan: A tool for identifying chimeric transcription in sequencing data, *Bioinformatics.* 27 (2011) 2903–2904. doi:10.1093/bioinformatics/btr467.
- [242] S. Tarazona, F. García-Alcalde, J. Dopazo, A. Ferrer, A. Conesa, Differential expression in RNA-seq: A matter of depth, *Genome Res.* 21 (2011) 2213–2223. doi:10.1101/gr.124321.111.
- [243] Y. Liu, J. Zhou, K.P. White, RNA-seq differential expression studies: More sequence or more replication?, *Bioinformatics.*

- 30 (2014) 301–304. doi:10.1093/bioinformatics/btt688.
- [244] M.A. Busby, C. Stewart, C.A. Miller, K.R. Grzeda, G.T. Marth, Scotty: A web tool for designing RNA-Seq experiments to measure differential gene expression, *Bioinformatics*. 29 (2013) 656–657. doi:10.1093/bioinformatics/btt015.
 - [245] S. Andrews, FastQC: A quality control tool for high throughput sequence data, *Bioinformatics*. (2010) 1. doi:citeulike-article-id:11583827.
 - [246] X. Yang, D. Liu, F. Liu, J. Wu, J. Zou, X. Xiao, et al., HTQC: a fast quality control toolkit for Illumina sequencing data., *BMC Bioinformatics*. 14 (2013) 33. doi:10.1186/1471-2105-14-33.
 - [247] Hannon-Lab, FASTX-Toolkit, (2010). http://hannonlab.cshl.edu/fastx_toolkit/ (accessed February 1, 2016).
 - [248] M. Martin, Cutadapt removes adapter sequences from high-throughput sequencing reads, *EMBnet.journal*. 17 (2011) 10. doi:10.14806/ej.17.1.200.
 - [249] A.M. Bolger, M. Lohse, B. Usadel, Trimmomatic: a flexible trimmer for Illumina sequence data, *Bioinformatics*. 30 (2014) 2114–2120. doi:10.1093/bioinformatics/btu170.
 - [250] E. Kopylova, L. No  , H. Touzet, SortMeRNA: fast and accurate filtering of ribosomal RNAs in metatranscriptomic data, *Bioinformatics*. 28 (2012) 3211–3217. doi:10.1093/bioinformatics/bts611.
 - [251] A. Oshlack, M.D. Robinson, M.D. Young, From RNA-seq reads to differential expression results., *Genome Biol*. 11 (2010) 220. doi:10.1186/gb-2010-11-12-220.
 - [252] A. Conesa, P. Madrigal, S. Tarazona, D. Gomez-Cabrero, A. Cervera, A. McPherson, et al., A survey of best practices for RNA-seq data analysis, *Genome Biol*. 17 (2016) 13. doi:10.1186/s13059-016-0881-8.
 - [253] M. Kostadima, R. Loos, RNA-seq: transcriptome assembly and differential expression, EMBL-EBI. (2012). https://www.ebi.ac.uk/training/online/sites/ebi.ac.uk.training.online/files/user/18/private/rnaseq_kostadima.pdf (accessed June 7, 2017).
 - [254] Y. Xie, G. Wu, J. Tang, R. Luo, J. Patterson, S. Liu, et al., SOAPdenovo-Trans: De novo transcriptome assembly with short RNA-Seq reads, *Bioinformatics*. 30 (2014) 1660–1666. doi:10.1093/bioinformatics/btu077.
 - [255] C. Trapnell, L. Pachter, S.L. Salzberg, TopHat: Discovering splice junctions with RNA-Seq, *Bioinformatics*. 25 (2009) 1105–1111. doi:10.1093/bioinformatics/btp120.
 - [256] M. Lawrence, W. Huber, H. Pag  s, P. Aboyoun, M. Carlson, R. Gentleman, et al., Software for Computing and Annotating Genomic Ranges, *PLoS Comput. Biol*. 9 (2013). doi:10.1371/journal.pcbi.1003118.
 - [257] Y. Liao, G.K. Smyth, W. Shi, The Subread aligner: Fast, accurate and scalable read mapping by seed-and-vote, *Nucleic Acids Res*. 41 (2013). doi:10.1093/nar/gkt214.
 - [258] N. Delhomme, I. Padioleau, E.E. Furlong, L.M. Steinmetz, easyRNASeq: a Bioconductor package for processing RNA-Seq data, *Bioinformatics*. in press (2012) in press. doi:10.1093/bioinformatics/bts477.
 - [259] S. Anders, P. Pyl, W. Huber, HTSeq--A Python framework to work with high-throughput sequencing data, *bioRxiv*. (2014).
 - [260] S. Anders, W. Huber, Differential expression analysis for sequence count data, *Genome Biol*. 11 (2010) R106. doi:10.1186/gb-2010-11-10-r106.
 - [261] B.M. Bolstad, B.M. Bolstad, R. a Irizarry, R. a Irizarry, M.   Strand, M.   Strand, et al., A comparison of normalization methods for high density oligonucleotide array data based on variance and bias, *Bioinformatics*. 19 (2003) 185–193. doi:10.1093/bioinformatics/19.2.185.
 - [262] J.H. Bullard, E. Purdom, K.D. Hansen, S. Dudoit, Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments., *BMC Bioinformatics*. 11 (2010) 94. doi:10.1186/1471-2105-11-94.
 - [263] M. Robinson, A. Oshlack, A scaling normalization method for differential expression analysis of RNA-seq data, *Genome Biol*. 11 (2010) R25. doi:10.1186/gb-2010-11-3-r25.
 - [264] C. Trapnell, B. a Williams, G. Pertea, A. Mortazavi, G. Kwan, M.J. van Baren, et al., Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation., *Nat. Biotechnol*. 28 (2010) 511–515. doi:10.1038/nbt.1621.
 - [265] M.A. Dillies, A. Rau, J. Aubert, C. Hennequet-Antier, M. Jeanmougin, N. Servant, et al., A comprehensive evaluation of normalization methods for Illumina high-throughput RNA sequencing data analysis, *Brief. Bioinform*. 14 (2013) 671–683. doi:10.1093/bib/bbs046.

- [266] M.D. Robinson, D.J. McCarthy, G.K. Smyth, edgeR: A Bioconductor package for differential expression analysis of digital gene expression data, *Bioinformatics*. 26 (2010) 139–140. doi:10.1093/bioinformatics/btp616.
- [267] J.C. Marioni, C.E. Mason, S.M. Mane, M. Stephens, Y. Gilad, RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays., *Genome Res.* 18 (2008) 1509–17. doi:10.1101/gr.079558.108.
- [268] L. Wang, Z. Feng, X. Wang, X. Wang, X. Zhang, DEGseq: an R package for identifying differentially expressed genes from RNA-seq data., *Bioinformatics*. 26 (2010) 136–8. doi:10.1093/bioinformatics/btp612.
- [269] U. Nagalakshmi, Z. Wang, K. Waern, C. Shou, D. Raha, M. Gerstein, et al., The transcriptional landscape of the yeast genome defined by RNA sequencing., *Science*. 320 (2008) 1344–9. doi:10.1126/science.1158441.
- [270] M.D. Robinson, G.K. Smyth, Moderated statistical tests for assessing differences in tag abundance., *Bioinformatics*. 23 (2007) 2881–7. doi:10.1093/bioinformatics/btm453.
- [271] Z. Fang, J.A. Martin, Z. Wang, Statistical methods for identifying differentially expressed genes in RNA-Seq experiments, *Cell Biosci.* 2 (2012) 26. doi:10.1186/2045-3701-2-26.
- [272] S. Anders, A. Reyes, W. Huber, Detecting differential usage of exons from RNA-seq data, *Genome Res.* 22 (2012) 2008–2017. doi:10.1101/gr.133744.111.
- [273] S. Zheng, L. Chen, A hierarchical Bayesian model for comparing transcriptomes at the individual transcript isoform level, *Nucleic Acids Res.* 37 (2009). doi:10.1093/nar/gkp282.
- [274] C. Trapnell, A. Roberts, L. Goff, G. Pertea, D. Kim, D.R. Kelley, et al., Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks., *Nat. Protoc.* 7 (2012) 562–78. doi:10.1038/nprot.2012.016.
- [275] C. Trapnell, D.G. Hendrickson, M. Sauvageau, L. Goff, J.L. Rinn, L. Pachter, Differential analysis of gene regulation at transcript resolution with RNA-seq., *Nat. Biotechnol.* 31 (2013) 46–53. doi:10.1038/nbt.2450.
- [276] Y. Shi, H. Jiang, rSeqDiff: detecting differential isoform expression from RNA-Seq data using hierarchical likelihood ratio test, *PLoS One*. 8 (2013) e79448. doi:10.1371/journal.pone.0079448.
- [277] Elena Daniela Aflorei, Application of a *Drosophila melanogaster* model to study Familial Isolated Pituitary Adenomas syndrome pathogenesis in vivo, Queen Mary, University of London, 2016.
- [278] M. Johnson, I. Zaretskaya, Y. Raytselis, Y. Merezhuik, S. McGinnis, T.L. Madden, NCBI BLAST: a better web interface., *Nucleic Acids Res.* 36 (2008). doi:10.1093/nar/gkn201.
- [279] M. Burrows; D. J. Wheeler, A block-sorting lossless data compression algorithm, (1994). <http://www.hpl.hp.com/techreports/Compaq-DEC/SRC-RR-124.pdf>.
- [280] Z. Khan, J.S. Bloom, L. Kruglyak, M. Singh, A practical algorithm for finding maximal exact matches in large sequence datasets using sparse suffix arrays, *Bioinformatics*. 25 (2009) 1609–1616. doi:10.1093/bioinformatics/btp275.
- [281] A. Dobin, C.A. Davis, F. Schlesinger, J. Drenkow, C. Zaleski, S. Jha, et al., STAR: ultrafast universal RNA-seq aligner, *Bioinformatics*. 29 (2013) 15–21. doi:10.1093/bioinformatics/bts635.
- [282] P. Flicek, M.R. Amodé, D. Barrell, K. Beal, K. Billis, S. Brent, et al., Ensembl 2014, *Nucleic Acids Res.* 42 (2014). doi:10.1093/nar/gkt1196.
- [283] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, et al., The Sequence Alignment/Map format and SAMtools, *Bioinformatics*. 25 (2009) 2078–2079. doi:10.1093/bioinformatics/btp352.
- [284] M. Love, W. Huber, S. Anders, Moderated estimation of fold change and dispersion for RNA-Seq data with DESeq2, *bioRxiv*. (2014).
- [285] R. a van den Berg, H.C.J. Hoefsloot, J. a Westerhuis, A.K. Smilde, M.J. van der Werf, Centering, scaling, and transformations: improving the biological information content of metabolomics data., *BMC Genomics*. 7 (2006) 142. doi:10.1186/1471-2164-7-142.
- [286] S. Dudoit, Y.H. Yang, M.J. Callow, T.P. Speed, Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments, *Stat. Sin.* 12 (2002) 111–139. #.
- [287] G. Dennis Jr, B.T. Sherman, D.A. Hosack, J. Yang, W. Gao, C.H. Lane, et al., DAVID: Database for Annotation, Visualization, and Integrated Discovery, *Genome Biol.* 4 (2003) P3. doi:10.1186/gb-2003-4-9-r60.
- [288] J.E. Rebers, L.M. Riddiford, Structure and expression of a *Manduca sexta* larval cuticle gene homologous to *Drosophila* cuticle genes, *J. Mol. Biol.* 203 (1988) 411–423. doi:10.1016/0022-2836(88)90009-5.
- [289] X. Guan, B.W. Middlebrooks, S. Alexander, S.A. Wasserman, Mutation of TweedleD, a member of an unconventional

- cuticle protein family, alters body shape in *Drosophila*., *Proc. Natl. Acad. Sci. U. S. A.* 103 (2006) 16794–9. doi:10.1073/pnas.0607616103.
- [290] H. Ogata, S. Goto, K. Sato, W. Fujibuchi, H. Bono, M. Kanehisa, KEGG: Kyoto encyclopedia of genes and genomes, *Nucleic Acids Res.* 27 (1999) 29–34. doi:10.1093/nar/27.1.29.
- [291] W.R. Bishop, R.M. Bell, Functions of diacylglycerol in glycerolipid metabolism, signal transduction and cellular transformation., *Oncogene Res.* 2 (1988) 205–18. <http://www.ncbi.nlm.nih.gov/pubmed/3285300>.
- [292] C. Kappen, M.A. Mello, R.H. Finnell, J.M. Salbaum, Folate modulates Hox gene-controlled skeletal phenotypes, *Genesis.* 39 (2004) 155–166. doi:10.1002/gene.20036.
- [293] C. Chang, C. Lin, LIBSVM: A Library for Support Vector Machines, *ACM Trans. Intell. Syst. Technol.* 2 (2011) 1–39. doi:10.1145/1961189.1961199.